WAS
8332

Volume 109
Number 2
Summer 2023

Journal of the WASHINGTON ACADEMY OF SCIENCES

Editor's Comments *K. Baclawski* ii
Board of Discipline Editors iii
Academy Board Members iv
Instruction to Authors v
Affiliated Institutions vi
The Cosmological Catastrophe *S. Howard* 1
Much Ado About Nothing *K. Baclawski* 13
MSVIPER: Robot Navigation *A. M. Roth et al* 27
Obituary: Albert H. Teich 59
Affiliated Societies and Delegates 60

ISSN 0043-0439 **Issued Quarterly at Washington DC**

Washington Academy of Sciences

Founded in 1898

Board of Managers

Elected Officers

President
Mahesh Mani

President Elect
Ram Sriram

Treasurer
David Torain

Secretary
Paul Arveson

Vice President, Administration
Anne Kornahrens

Vice President, Membership
Mala Ramaiah

Vice President, Junior Academy
Vijay Kowtha

Vice President, Affiliated Societies
Mei Sun

Members at Large
Terry Longstreth
Michael Cohen
George Dimitoglou
Judy Staveley
Guru Venkataramani

Past President
Lynnette Madsen

Affiliated Society Delegates

Shown on back cover

Editor of the Journal
Kenneth Baclawski

Journal of the Washington Academy of Sciences (ISSN 0043-0439)

Published by the Washington Academy of Sciences
email: editor@washacadsci.org
website: www.washacadsci.org

The Journal of the Washington Academy of Sciences

The *Journal* is the official organ of the Academy. It publishes articles on science policy, the history of science, critical reviews, original science research, proceedings of scholarly meetings of its Affiliated Societies, and other items of interest to its members. It is published quarterly. The last issue of the year contains a directory of the current membership of the Academy.

Subscription Rates

Members, fellows, and life members in good standing receive the Journal free of charge. Subscriptions are available on a calendar year basis, payable in advance. Payment must be made in US currency at the following rates.

US and Canada	$30.00
Other Countries	$35.00
Single Copies (when available)	$15.00

Claims for Missing Issues

Claims must be received within 65 days of mailing. Claims will not be allowed if non-delivery was the result of failure to notify the Academy of a change of address.

Notification of Change of Address

Address changes should be sent promptly to the Academy Office. Notification should contain both old and new addresses and zip codes.

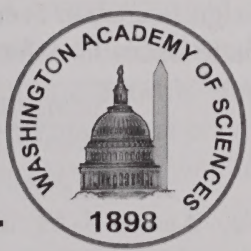
Postmaster:
Send address changes to WAS, Rm 455,
1200 New York Ave. NW
Washington, DC 20005

Academy Office
Washington Academy of Sciences
Room 455
1200 New York Ave. NW
Washington, DC 20005
Phone: (202) 558-0938

MCZ LIBRARY
FEB 15 2024
HARVARD UNIVERSITY

Volume 109
Number 2
Summer 2023

Journal of the WASHINGTON ACADEMY OF SCIENCES



Editor's Comments *K. Baclawski*....ii
Board of Discipline Editors....iii
Academy Board Members....iv
Instruction to Authors....v
Affiliated Institutions....vi
The Cosmological Catastrophe *S. Howard*....1
Much Ado About Nothing *K. Baclawski*....13
MSVIPER: Robot Navigation *A. M. Roth et al*....27
Obituary: Albert H. Teich....59
Affiliated Societies and Delegates....60

ISSN 0043-0439 **Issued Quarterly at Washington DC**

EDITOR'S COMMENTS

The problem of the cosmological constant is one of the most dramatic examples of a discrepancy between theory and observation in science today. Theoretical estimates range from 50 to more than 120 orders of magnitude greater than the observed values. The disagreement is so large that it has been called a catastrophe. Sethanne Howard has presented an informative introduction to the history and current state of affairs of this outstanding problem in physics in the leading article of this issue.

The cosmological constant is the energy density of the vacuum. The idea that there could be a physical or conceptual state that consists of nothing has been debated by philosophers since antiquity. Does it even make sense for nonexistence to exist? One might dismiss such speculations as having no significance in everyday life, yet it turns out that there are important pragmatic issues for what nothingness means and how one deals with it. Sethanne Howard's article covers the problem of the physical vacuum. The second article surveys conceptual nothingness and presents many examples of how it matters.

In the last article, Aaron Roth and his team developed an innovative technique for improving robot navigation by using reinforcement learning to construct a decision-tree policy. They demonstrate that their technique significantly improves the ability of mobile robots to navigate in dynamically changing environments and on complex, uneven terrain.

Please send any news about yourselves as well as your comments on papers, suggestions for articles, and ideas for what you would like to see in the Journal to editor@washacadsci.org.

Kenneth Baclawski

Journal of the Washington Academy of Sciences

Editor Kenneth Baclawski editor@washacadsci.org

Board of Discipline Editors

The *Journal of the Washington Academy of Sciences* has a Board of Discipline Editors representing many scientific and technical fields. The members of the Board of Discipline Editors are affiliated with a variety of scientific institutions in the Washington area and beyond — government agencies such as the National Institute of Standards and Technology (NIST); universities such as Georgetown; and professional associations such as the Institute of Electrical and Electronics Engineers (IEEE).

Anthropology	Emanuela Appetiti	eappetiti@hotmail.com
Astronomy	Sethanne Howard	sethanneh@msn.com
Computer Science	George Dimitoglou	dimitoglou@hood.edu
Health	Robin Stombler	rstombler@auburnstrat.com
History of Medicine	Alain Touwaide	atouwaide@hotmail.com
Operations Research	Michael Katehakis	mnk@rci.rutgers.edu
Psychology	Gary Berg-Cross	gbergcross@gmail.com
Science Education	Jim Egenrieder	jim@deepwater.org

Academy Board Members

President	Mahesh Mani
President-Elect	Ram Sriram
Secretary	Paul Arveson
Treasurer	David Torain
Vice President for Membership Affairs	Mala Ramaiah
Vice President, Administrative Affairs	Anne Kornahrens
Vice President, Affiliated Societies	Mei Sun
Vice President, Junior Academy	Vijay Kowtha
At-Large Board Members	Terry Longstreth (2023-2024) Michael Cohen (2023-2024) George Dimitoglou (2023-2024) Judy Staveley (2022-2025) Guru Venkataramani (2023-2026)
Immediate Past President	Lynnette Madsen
Journal Editor	Kenneth Baclawski
Affiliated Societies Representatives	Michael Cohen Sethanne Howard

Instructions to Authors

1. Deadlines for quarterly submissions are:

   Spring – February 1 Fall – August 1
   Summer – May 1 Winter – November 1

2. Draft Manuscripts using a word processing program (such as MSWord), not PDF. We do not accept PDF manuscripts.
3. Papers should be 6,000 words or fewer. If there are seven or more graphics, reduce the number of words by 500 for each graphic.
4. Include an abstract of 150-200 words.
5. Use Times New Roman, font size 12.
6. Include two to three sentence bios of the authors.
7. Graphics must be easily resizable by the editor to fit the Journal's page size. Reference the graphic in the text.
8. Use endnotes or footnotes. The bibliography may be in a style considered standard for the discipline or professional field represented by the paper.
9. Submit papers as email attachments to the editor at editor@washacadsci.org.
10. Include the author's name, affiliation, and contact information – including postal address. Membership in an Academy-affiliated society may also be noted. It is not required.
11. Manuscripts are peer reviewed and become the property of the Washington Academy of Sciences.
12. There are no page charges.

Washington Academy of Sciences
Affiliated Institutions

National Institute for Standards & Technology (NIST)
Meadowlark Botanical Gardens
The John W. Kluge Center of the Library of Congress
Potomac Overlook Regional Park
Koshland Science Museum
American Registry of Pathology
Living Oceans Foundation
National Rural Electric Cooperative Association (NRECA)

The Cosmological Catastrophe

Sethanne Howard
USNO/retired

Abstract

The cosmological catastrophe remains an outstanding problem in cosmology. Science still lacks a way to combine the General Theory of Relativity (science writ large) with Quantum Mechanics (science writ small). I discuss here how this catastrophe got started and where it stands now.

Introduction

Before we define the catastrophe let us start with gravity. Sir Isaac Newton defined gravity as a force that permeates the universe. The gravitational force that drops an apple from a tree to the ground is the same gravitational force that lets the Moon orbit the Earth. It was a groundbreaking realization. Astronomers could now map out celestial motions especially in the Solar System. Within a hundred years we had added Uranus to the catalog of planets in our Solar System. Using celestial mechanics (forces in the universe) astronomers defined an orbit for Uranus; but it had small wobbles in its orbit. What could cause wobbles? Another gravitational force can do it, and that led eventually to the discovery of the planet Neptune. Newton's laws worked well.

One might ask if there are wobbles in Neptune's orbit, thus implying another planet even farther out. There are very tiny wobbles in Neptune's orbit. Astronomers have studied these wobbles in great detail and have shown that the wobbles are too small to support the idea of another planet (personal communication).

Then several things happened around the beginning of the twentieth century that challenged Newton.. Einstein forecast the field of quantum mechanics with the discovery of the photoelectric effect. The photoelectric effect is the emission of electrons when electromagnetic radiation, such as light, hits a material. Einstein proposed that a beam of light acts not as a wave propagating through space, but instead as a swarm of discrete energy packets, known as photons—term coined by Gilbert N. Lewis in 1926. It

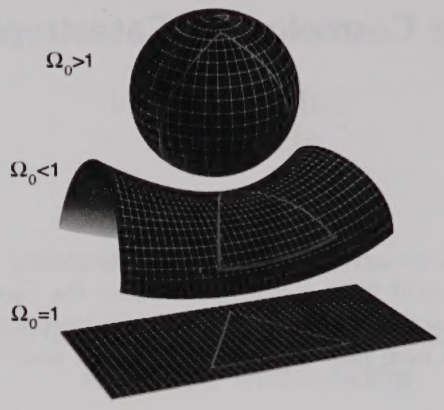


Figure 1: Three maps of the curvature of space; from top to bottom the curvature is negative, positive and zero (flat). Source: Wikipedia.

was for this work that Einstein received the Nobel Prize. Light is both a wave and a particle. The speed of light, c, was measured. It was not infinite as Newton believed. It has a finite value of c = 299,792,458 meters per second.

Einstein also developed the Special Theory of Relativity (STR) where space and time are no longer independent of each other, and the speed of light is a constant. What was a Euclidean metric with $ds^2 = dx^2 + dy^2 + dz^2$ in classical physics is now $ds^2 = dt^2 - dx^2 - dy^2 - dz^2$ in STR. See Howard (2011) for an explanation of a metric. Metrics are necessary for Einstein's work. Various fields of mathematics developed. One of them is non-Euclidean geometry. Euclidian geometry is our old familiar (x, y, z) flat system. However, there are mathematical spaces that are not flat; they curve. Figure 1 shows three maps of space curvature. The bottom image represents a Euclidean (flat) space. The middle image represents a negatively curved space (looks a bit like a saddle). The top image represents a positively curved space (like the surface of the Earth). Superimposed on each space is a triangle. Notice how its shape varies with the curvature. In a flat space the angles in the triangle add to 180°. In the other two spaces the angles in the triangle do not add to 180°.

A positively curved space is a closed space. When you walk in a "straight" line you will return to where you started. A negatively curved

space is an open space. When you walk in a "straight" line you will not return to where you started.

The mathematical word for "straight" line is geodesic. The geodesics are drawn as a grid in each space in Figure 1.

Every spacetime has its metric, a method of measuring distance between two points in that spacetime. For the Special Theory of Relativity the metric is

$$ds^2 = dt^2 - dx^2 - dy^2 - dz^2$$

where ds^2 represents the distance in Euclidean spacetime.

Also put forward in the late 19^{th} century was Riemann geometry. It deals with a broad range of geometries, each with its metric. Every smooth space has a Riemannian metric. A 'space' in differential geometries is called a manifold. So every smooth manifold has a Riemannian metric. In differential geometry a differentiable manifold is a space which is locally similar to a Euclidean space.

When one moves from three to four dimensions one uses pseudo-Riemannian manifolds. In this case the metrics are everywhere nondegenerate. See Figure 2 for an example.

An n-dimensional differentiable manifold is a generalization of n-dimensional Euclidean space. In a manifold it may only be possible to define coordinates *locally*.

Einstein's Special Theory of Relativity uses a four-dimensional space. It has the metric defined above.

There is one more late 19^{th} century scientific result to discuss. It is a resolved astronomical catastrophe known as the ultraviolet catastrophe. Maxwell codified electromagnetic radiation when he produced what are now called Maxwell's Laws. He treated electromagnetic radiation as a wave. There are four laws that describe the behavior of electromagnetic radiation. You can look up the meaning of Maxwell's Laws in any book on electromagnetic theory.

Blackbody radiation is the thermal electromagnetic radiation within, or surrounding, a body in thermodynamic equilibrium with its environment. In other words they are at the same temperature. Scientists

Figure 2: Projecting a sphere to a plane using zero dimensions
Source: Wikipedia.

can make a blackbody. They use an insulated enclosure which is in thermal equilibrium. It contains blackbody radiation and will emit it through a hole made in its wall, provided the hole is small enough to have a negligible effect upon the equilibrium. Astronomers use blackbody sources to measure the thermal energy distribution of a star.

The term blackbody was introduced by Gustav Kirchhoff in 1860. Blackbody radiation is also called thermal radiation. All baryonic matter emits electromagnetic radiation when it has a temperature above absolute zero. Baryonic matter in thermal equilibrium with its environment emits blackbody radiation.

The Cosmic Background Explorer (COBE) mission operated between 1989 and 1993. The goal was to measure the cosmic microwave (blackbody) radiation left over from the Big Bang.

The announcement of this result at a meeting of the American Astronomical Society led to a standing ovation from the audience, only the second such to occur over the lifetime of the Society. This Figure shows the shape of a blackbody spectrum.

The problem comes when one considers the distribution of blackbody radiation versus temperature.

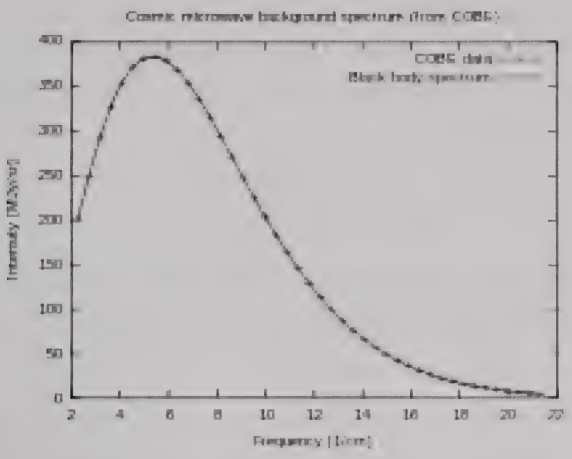


Figure 3: The nearly perfect match to a blackbody spectrum as measured by COBE. The error bars are smaller than the width of the line. Energy response is plotted versus frequency. Source: Wikipedia.

According to the classical theory of electromagnetic radiation, if each Fourier mode of the equilibrium radiation (in an otherwise empty cavity with perfectly reflective walls) is considered as a degree of freedom capable of exchanging energy, then, according to the equipartition theorem of classical physics, there would be an equal amount of energy in each mode. Since there are an infinite number of modes, this would imply infinite heat capacity, as well as a nonphysical spectrum of emitted radiation that grows without bound with increasing frequency, a problem known as the ultraviolet catastrophe.

In long wavelengths this effect is not very noticeable; however, in the shorter wavelengths the response of the spectrum increases ad infinitum. This is a problem since there is no infinite energy response seen.

The blackbody response is shown in Figure 3. Note that the response turns over and returns to zero. It does not continue to increase in the short (ultraviolet) wavelengths as expected from the classical theory.

The ultraviolet catastrophe, also called the Rayleigh–Jeans catastrophe, was the prediction that an ideal blackbody at thermal equilibrium would emit an unbounded quantity of energy as wavelength decreased into

the ultraviolet range. The term "ultraviolet catastrophe" was first used in 1911 by Paul Ehrenfest, but the concept originated with the 1900 statistical derivation of the Rayleigh–Jeans law. That law was empirically determined. Its formula is:

$$B_\lambda(T) = \frac{2ck_BT}{\lambda^4}$$

The spectral radiance is inversely proportional to the fourth power of the wavelength and directly proportional to the temperature.

A resolution was sorely needed.

In 1900 Max Planck derived the form for the intensity of the spectral distribution function by making some assumptions. He assumed that radiation can be absorbed or emitted in discrete packets called quanta or:

$$E = h\nu = h\frac{c}{\lambda}$$

where h is Planck's constant. Using this assumption we get the correct form of the spectral energy distribution:

$$B_\lambda(T) = \frac{2hc^2}{\lambda^5}\frac{1}{\exp\left(\frac{hc}{\lambda k_BT}\right) - 1}$$

This ought to look familiar. It avoids the ultraviolet catastrophic, produces the correct blackbody curve, and all is well.

In 1915 Einstein used Riemannian geometry to build the General Theory of Relativity (GTR) where mass now can affect light – a new concept. His GTR replaces that of Newton and applies to objects on the cosmic scale. Gravity is no longer a force; it is embedded in the geometry of spacetime. His equation for a geometric theory of gravity is:

$$G^\nu_\mu + \Lambda g^\nu_\mu = \frac{8\pi G}{c^4}T^\nu_\mu$$

where this is an equation with 16 dimensions in tensor form. The left side of the equation gives the geometry of the spacetime. The right side of the equation gives the energy due to the mass.

Mass tells spacetime how to move, and spacetime tells the mass how to move. G^{ν}_{μ} is the Einstein tensor – the curvature in a Riemannian manifold. Λ is the cosmological constant – initially inserted to make a static universe, and later removed by Einstein. Remember this one because it will be important. g^{ν}_{μ} is the metric tensor – the Riemannian manifold has a metric. T^{ν}_{μ} is the stress energy tensor representing the mass.

Definition of Cosmological Catastrophe

Now that we have the official definition of gravity (as given by Einstein, which replaced Newton's definition of gravity) we can define the *cosmological catastrophe*. It has to do with the value of Λ. Einstein first inserted Λ to keep the universe from expanding or contracting – he wanted a static universe. But a static universe is unstable, and shortly thereafter Georges Lemaître showed that the universe is expanding. By 1931 Einstein accepted these results and set Λ to zero where it stayed for most of the 20^{th} century.

Yet astronomers wanted to determine the value of Λ. There are two ways to get a value for Λ. One way uses the value for Λ from astronomical observations. The other way is to use the theoretical value for Λ from quantum theory. One would hope the two ways would give the same value; however, it was not to be. The two values differ between 50 and 120 orders of magnitude – a difference too large to accept, a veritable catastrophe in cosmology.

Λ is taken to be the lowest possible energy density which occurs at the zero-point energy. Einstein had set this value to zero. In quantum mechanics, however, the vacuum state is associated with the zero-point energy, and this zero-point energy (equivalent to the lowest possible energy state) has measurable effects. In the laboratory it may be detected as the Casimir effect. The Casimir effect (or Casimir force) is a physical force acting on the macroscopic boundaries of a confined space which arises from the quantum fluctuations of a field.

Let us see how the two methods work.

Dark Energy

In 1998 two teams of astrophysicists carried out measurements on distant Type Ia supernovae which showed that the speed of galaxies' recession in relation to the Milky Way increases over time (Perlmutter, 1999; Riess, 1998). The universe is in accelerated expansion, which requires having a strictly positive Λ. The universe thus would contain a mysterious dark energy producing a repulsive force that over-counterbalances the gravitational braking produced by the matter contained in the universe. See Figure 4.

Their work showed the universe not only is expanding, but it is also accelerating. The matter in the universe is insufficient to slow down the universe to a leisurely expansion. A positive vacuum energy density resulting from a positive Λ implies a negative pressure, and *vice versa*. If the energy density is positive, the associated negative pressure will drive an accelerated expansion of the universe, as observed.

We do not know what comprises dark energy. The cosmological constant Λ is presumed to be an enigmatic from of matter or energy that acts in opposition to gravity and is thought to be equivalent to dark energy. Nobody really knows what Λ is exactly, but it is required in cosmological equations to reconcile theory with our observations of the universe. It is the simplest explanation for dark energy, and it is used in the current standard model of cosmology known as the ΛCDM (Lambda Cold Dark Matter) model. The cosmological constant has the same effect as an intrinsic energy density of the vacuum, ρ_{vac} (and an associated pressure). Λ represents dark energy in empty space or vacuum energy. It has the dimension of length^{-2}.

Observations - The Planck Mission

What are the astronomical observations? The Planck mission was a space observatory operated by the European Space Agency (ESA) from 2009 to 2013. It is named after Max Planck the scientist who resolved the ultraviolet catastrophe. One of the mission objectives was to measure the value of Λ. It was a highly successful mission that provides the cosmological data used currently. The James Webb Space Telescope will also produce these data, but it is too early in its mission to draw conclusions.

On 5 July 2010 the Planck mission delivered its first all-sky im-

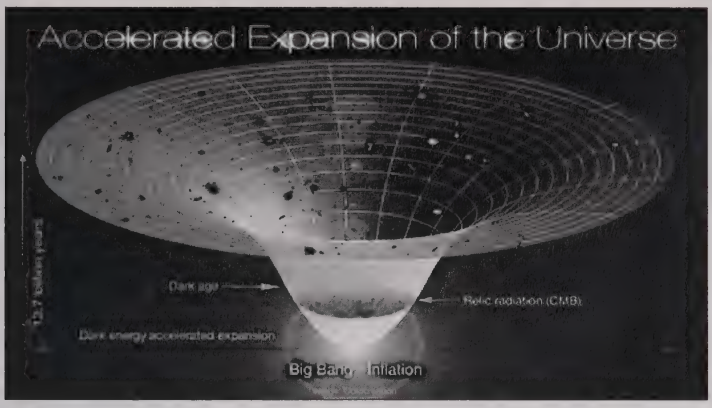


Figure 4: Accelerating universe. Source: Wikipedia.

age. The first public scientific result of Planck is the Early-Release Compact-Source Catalogue, released during the January 2011 Planck conference in Paris. The Hubble Constant $H_0 = 66.88 \pm 0.92$km sec^{-2}Mpc^{-2}, $\Omega_\Lambda = 0.679 \pm 0.013$. Ω_Λ is the ratio between Λ and the critical density of the universe, the tipping point for a sufficient density to stop the universe from expanding forever.

$$\Lambda = 1.1056 \times 10^{-52} m^{-2}$$

Using the dimension of Planck length the cosmological constant has the value:

$$\Lambda = 2.888 \times 10^{-122} l_P^{-2}$$

where the Planck length is $1.616255(18) \times 10^{-35}$ m, a fundamental constant. Λ is a positive, although very small number.

According to the team the universe is 13.798 ± 0.037 billion years old, and contains $4.82 \pm 0.05\%$ ordinary matter, $25.8 \pm 0.4\%$ dark matter and $69 \pm 1\%$ dark energy. See Figure 5.

The value is based on recent measurements of vacuum energy density, $\rho_{vac} = 5.96 \times 10^{-27}\ kg/m^3 = 5.3566 \times 10^{-10}\ J/m^3 = 3.35\ GeV/m^3$. However, there is work circulating that proposes that the cosmological principle is no longer true in the late universe and that the metric breaks down, so it is possible that observations usually attributed to an accelerating uni-

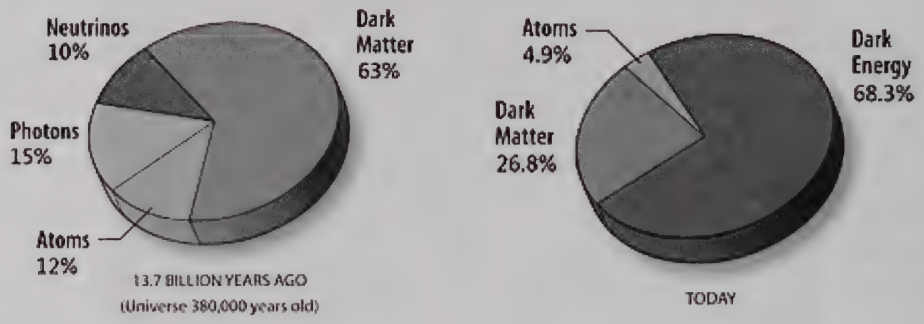


Figure 5: Distribution of matter and energy then and now - Source: Wikipedia.

verse are simply a result of the cosmological principle not applying in the late universe. We will not use this work here.

The cosmological principle, as in the case for all models that use the Friedmann–Lemaître–Robertson–Walker (FLRW) metric, is in most respects the simplest solution. FLRW is a metric based on the exact solution of the Einstein field equations of general relativity. The metric describes a homogeneous, isotropic, expanding universe. Its generic form is:

$$-c^2dr^2 = -c^2dt^2 + a(t)^2d\Sigma^2$$

where Σ ranges over a 3-dimensional space of uniform curvature.

Thus, the ΛCDM model, the current standard model of cosmology which uses the FLRW metric, includes the cosmological constant, which is measured to be on the order of $10^{-52}m^{-2}$.

Note that this value changes over time: The critical density changes with cosmological time, but the energy density due to the cosmological constant remains unchanged throughout the history of the universe. This is because the amount of dark energy increases as the universe grows but the amount of matter does not.

Prediction from Quantum Theory

Quantum theory applies to the very small – on the nuclear scale. According to quantum field theory, which underlies modern particle physics, empty space is defined by the vacuum state, which is composed of a col-

lection of quantum fields. All these quantum fields exhibit fluctuations in their ground state (lowest energy density) arising from the zero-point energy present everywhere in space.

Even a vacuum, supposedly corresponding to empty spacetime, is full of physical activity in the form of energy fluctuations. These fluctuations give rise to particles popping out, only to disappear shortly afterwards. This may sound like a mathematical trick, but such particles have been spotted in countless experiments. One estimate, made by Jérôme Martin of the Institut d'Astrophysique de Paris in 2012, placed the expected theoretical vacuum energy scale around 10^8 GeV^4, for a difference from the Planck value for Λ of about 55 orders of magnitude.

A common assumption is that the cosmological constant is equivalent to the vacuum density. No theory supports this assumption. No data support this assumption. Nevertheless it is a common assumption with supporting statements.

Another ratio that is used by scientists is the equation of state, usually denoted w, which is the ratio of pressure that dark energy puts on the universe to the energy per unit volume. This ratio is $w = -1$ for the cosmological constant used in the Einstein equations. The value $w = -1.028 \pm 0.032$, measured by the Planck Collaboration (2018) is consistent with -1, assuming w does not change over cosmic time.

If the universe is described by an effective local quantum field theory down to the Planck scale, then we would expect a cosmological constant of the order of 1 in reduced Planck units. The measured cosmological constant is smaller than this by a factor of 10^{120} (as described above). At the Planck scale the predictions of the Standard Model (ΛCDM), quantum field theory, and general relativity no longer apply, and quantum effects of gravity dominate.

Conclusion

There are two methods to measure the value of Einstein's cosmological constant Λ: one involves actual data found by observing the sky, another one by making assumptions in quantum field theory. They differ by an unacceptable amount of up to about 60 orders of magnitude. As an old fashioned observational astronomer I tend to favor the value that

involves actual data. As a scientist who likes quantum mechanics I can see what has been done to determine Λ from quantum field theory. Hence we have a cosmological catastrophe, and the brain has to deal with two wildly different aspects of the same thing. The answer lies in the marriage of quantum mechanics with gravity, something yet to be accomplished.

If we ask where did spacetime come from? We can turn back the clock further and further, until we reach the "Planck epoch" – a period so early in the universe's history that our best theories of physics break down. This era occurred only one ten-millionth of a trillionth of a trillionth of a trillionth of a second after the Big Bang. At this point space and time became subject to quantum fluctuations. Scientists ordinarily work separately with quantum mechanics, which explains the microworld of particles, and with general relativity, which applies on large, cosmic scales. But to understand the Planck epoch we need a complete theory of quantum gravity, merging the two.

It is a good question to which we do not have the answer.

References

Howard, S. (2011). *Journal of the Washington Academy of Sciences*, **97**(2): 1.

Perlmutter, S. et al, (1999). *Astrophysical Journal*, **517**: 565.

The Planck Collaboration (2020). "Planck 2018 results. VI. Cosmological parameters," *Astronomy and Astrophysics.* **641**: A6.

Riess, A. et al. (1998). *Astronomical Journal*, **116**: 1009.

Dr. Sethanne Howard is an astronomer retired from the US Naval Observatory as Chief of the Nautical Almanac Office. She worked at many places including Program Manager at the National Science Foundation for extragalactic astronomy and cosmology.

Much Ado About Nothing

A brief survey of the concept of nothingness from antiquity to modern systems

Kenneth Baclawski
Washington Academy of Sciences

Abstract

Whether it is called "nothing," "no value," "vacuum," "empty" or "zero" the concept of *nothing* has been a matter of philosophical as well as pragmatic disagreement since antiquity and continuing today.

Introduction

THE CONCEPT OF NOTHING, whether the abstract notion of nonexistence or the physical vacuum, has been debated since at least the fifth century BCE. The related notion of *zero* has an even longer history. One might think that such issues would have been settled long ago, yet in many ways the opposite is the case. The properties of the physical notion of a vacuum is currently a major unsolved problem. While the mathematical notions of zero and the empty set are well-established, modern computer programming and data languages differ markedly in how they deal with nothingness. The differences have a significant effect on how such languages behave when encountering nothingness, which is a continuing source of confusion. In this survey we outline some of the history of nothingness. We then discuss the main notions of nothingness in modern data and programming languages. Finally, we list some of the many explanations for why one might have nothing.

History

There are two distinct uses of zero. The first is for a place-holder in positional numbering systems when a place has no value. The second is for the absence of a quantity. The Sumerians had a base 60 positional numbering system as early as 2500 BCE, and used a space for a position with no value, but did not have a symbol for zero. By 1770 BCE, the Egyptians had a symbol for zero in accounting texts (Gheverghese, 2011), but their numbering system was not positional (O'Connor and Robertson, 2000). The Babylonians inherited the Sumerian positional numbering sys-

tem and subsequently introduced a symbol for a position with no value, but never used it alone. The ancient Greeks had neither a symbol for zero nor a positional numbering system, in spite of being aware of Babylonian mathematics (Wallin, 2002). The reluctance of the ancient Greeks to use zero may be due to their philosophical arguments about the nature of existence and the vacuum. This reluctance continued well into subsequent civilizations, including Medieval Europe. It wasn't until around 650 CE that Brahmagupta in India formalized arithmetic operations using zero (Wallin, 2002).

One of the earliest Greek philosophers to discuss the concept of nothingness was Parmenides at around 500 BCE. At this time nothingness meant nonexistence and included the physical void or vacuum. Parmenides argued that the very act of speaking about nothing implies that it exists, which contradicts the definition of nothing as nonexistence (Russell, 1995). On the other hand, as was pointed out by Leucippus, the co-founder of atomism with his student Democritus, one can only have motion if there is a void through which matter moves. It then follows that there must be a void between the atoms. Little is known about Leucippus, and there is a considerable controversy about whether Leucippus ever existed, starting with Epicurus at around 400 BCE and continuing to modern times (Skordoulis & Koutalis, 2013, p. 467; Graham, 2008, pp. 333–335). If there was no such philosopher as Leucippus, we have the ironic situation that the first person to speculate about the existence of nonexistence did not exist.

Unfortunately, the early Greek philosophers did not possess a technology that was adequate for experimentally testing the speculations of the atomists so the theory was not generally accepted at the time. Hero of Alexandria tried unsuccessfully to create an artificial vacuum in the first century CE, and partial vacuums were not demonstrated until the 17^{th} century. Extremely high artificial vacuums can now be achieved, and outer space is an even more extreme vacuum than artificial vacuums, but even if all particles of matter were removed from a small volume, there would still be photons and neutrinos, as well as dark energy, virtual particles, and other aspects of the quantum vacuum. While a perfect vacuum is not currently regarded as being physically realizable, it is meaningful to speak of the lowest possible energy state of physical space, which is called the quantum vacuum, zero-point field or vacuum state. The properties of

the vacuum state is currently a major unsolved problem which has been very well covered by Sethanne Howard in another article in this same issue (Howard, 2023).

As with the physical notion of a vacuum, the ontological notion of nonexistence is also an unsolved problem, or at least a problem that remains controversial. After the Greeks, many philosophers attempted to understand nothingness, sometimes giving the notion a religious or spiritual interpretation. John the Scot (c. 815–877) identified nothingness with God (Russell, 1995, pp. 396-401). Hegel in his *Science of Logic* asserted that "pure nothing" is the same as "pure being," but he then contradicted this assertion in the next section by asserting, "But it is equally true that they [pure nothing and pure being] are not undistinguished from each other, that, on the contrary, they are not the same..." Hegel then proceeds to resolve the contradiction by introducing the relationship of "becoming" between pure nothing and pure being. To Hegel pure being is closely related to Geist, which can be roughly translated into three English meanings: ghost, spirit, and mind or intellect. (Hegel, 1812-1816, §§ 132-134).

Modern philosophers have continued to grapple with nothingness. In 1929 Martin Heidegger gave his Freiburg inaugural lecture entitled "What is Metaphysics?" (Heidegger, 1929). In this lecture he announced "Das Nichts selbst nichtet" (Cartlidge, n.d.). Understanding this or translating it to English is problematic because Heidegger commonly invented new words to help convey his ideas. Unfortunately, doing this has made it difficult to understand his intentions not only in his native language but also in translations to other languages. The problematic word in Heidegger's announcement is "nichtet," which can be interpreted as the verb form of the word "nicht" ("not" in English). As there is no such verb in English, it has been variously translated as "annihilates," "nothings" or "noths," where the last one is a back-formation obtained from the word "nothing" when regarded as being the gerund of a verb. This leads to the speculation that Heidegger was attempting subtle humor by asserted the existence of nonexistence using a nonexistant word, but that seems unlikely. In any case, one can translate the announcement variously as "Nothingness itself annihilates," "Nothingness itself nothings" or "Nothingness itself noths."

Here are a few quotes from Heidegger's lecture as translated by

Goth (2013). To improve readability, hyphenated terms in the translation were changed to common English words, e.g., "no-thing" was changed to "nothingness."

> But why do we trouble ourselves about this nothingness? In fact, nothingness is indeed turned away by science and given up [on] as the null and void [das Nichtige]. But if we give up nothingness in such a way, do we not indeed accept it? But can we talk about an acceptance if we accept nothing [nichts]? Yet maybe all this back and forth has already turned into empty verbal wrangling. Science must then renew its seriousness and assert its soberness in opposition to this, so that it has only to do with being [um das Seiende geht].

Note how Heidegger appears to be harkening back to the argument of Parmenides. Heidegger then repeats the argument of Parmenides in various ways, as for example the following:

> Accordingly, every answer to this question is impossible from the outset. For it necessarily starts out in the form: nothingness "is" this or that. Question and answer alike are themselves just as nonsensical with respect to nothingness.

Finally, he announces "Das Nichts selbst nichtet" but makes it clear that the annihilation is not an occurrence of some sort whereby something else annihilates or negates. Nothingness does the annihilation on its own.

This soon earned Heidegger fame as a purveyor of metaphysical nonsense (Inwood, 1999). Nevertheless, Heidegger is often considered to be among the most important and influential philosophers of the 20^{th} century. One interpretation of Heidegger's announcement is that in spite of the self-contradictory aspects of nothingness as a concept, one can nevertheless regard it as have its own kind of existence.[1] In any case, I am reluctant to claim that Heidegger's announcement is not unmeaningless, since another philosopher subsequently developed this idea into a rich body of popular work.

[1] I suspect that Heidegger would disagree with this interpretation.

Jean-Paul Sartre was one of the many philosophers who were influenced by Heidegger. Sartre is regarded as being the most prominent existential philosopher, and his principal philosophical work is *Being and Nothingness* (Sartre, 1943), an obvious reference to Heidegger's principal work *Being and Time* (Heidegger, 1927). As the title suggests, nothingness is central to Sartre's work. Beginning in the first chapter, Sartre points out that nothingness is something we experience as part of reality. The absence of a family member or a lack of money are not just subjective nothings. Starting from this thesis, Sartre developed a wide range of topics such as consciousness, perception, social philosophy, self-deception, psychoanalysis, and the question of free will. Sartre was a prolific writer having written novels, short stories, plays, screenplays, autobiographical works as well as philosophical and other essays. His work is richly symbolic and clearly shows its basis in his philosophy.

There have been many other philosophers who have discussed nothingness, but the examples in this section should give one an inkling of the kinds of arguments that have been presented. None of the arguments have been entirely refuted. Indeed, the argument of Parmenides can be used to assert that a category is meaningful only if it potentially could have members. For example, one could argue that the category of unicorns is not meaningful, although the category of articles about unicorns is meaningful. On the other hand, one could allow for the existence of nothingness as asserted by Hegel, Heidegger and Sartre. We now give some examples of both approaches in modern systems.

Nothingness in Information Systems

Modern computer languages differ in how they deal with nothingness. The main distinction is whether nothingness should be regarded as a special kind of value (as asserted by Heidegger and Sartre) or should be no value at all (as suggested by Parmenides). The choice between these two is neither trivial nor obvious, and there are important advantages and disadvantages for the two choices. Specifically, most major programming languages have chosen to treat nothingness as a special value, while data languages, such as the standard relational database language SQL, have generally chosen to treat nothingness as the absence of a value.

Programming Languages

Historically, a notion of nothingness in programming languages was needed when programmers began to symbolically specify addresses (locations) of data in memory. There needed to be some default address when the symbol does not specify any data, for example, because the data has not yet been allocated a location in memory. Originally, this default address was the address 0. The C language still allows one to specify the default address with the digit 0 even though the actual default machine address need not be an address all of whose bits are 0. Other programming languages use names such as "null," "NULL" or "nil." Even in C it is now considered preferable to use the NULL symbol rather than the digit 0. Regardless of the specific name used, they are called null pointers or null references. Tony Hoare invented the null reference in 1965 as part of the ALGOL W language (Hoare, 2009).

Null pointers are commonly used to represent conditions such as the end of a sequence of actions having unknown length or the result of an action that did not produce any data. An important requirement for nulls is that one can test for them by a simple comparison. In particular, if two program variables that both have the null value are compared, then the result is TRUE.

Accessing the data referenced by a data pointer is known as "dereferencing." Because a null pointer does not point to a meaningful object, attempting to dereference it may cause a run-time error or program crash. This is called a null pointer exception. It is one of the most common types of software weakness. Because of this Tony Hoare referred to his invention of the null reference as a "billion dollar mistake" (Hoare, 2009).

While dereferencing a null reference is an exception in most programming languages, this is not always the case. For example, in Lisp the default value is called nil, and it can be used like any other value with well defined results and no exceptions. Python is similar to Lisp except that nil is called "None." However, in either of these languages the disadvantage is that if a use of nil/None was actually an error, then one could get into an infinite loop, which is arguably worse than a program crash. Objective-C is another language where dereferencing nil is not an exception, although nil is not treated as being an ordinary value. Unlike Lisp, dereferencing nil

is ignored in Objective-C. As with Lisp, erroneous uses of nil could result in an infinite loop.

For most programming languages where deferencing a null pointer causes an exception, one can catch the exception and take appropriate corrective action. Arguably it is better to generate an exception for an invalid dereference operation than to process it or to ignore it, which will just postpone the problem and make finding the error much more difficult.

Relational Database Languages

The notion of nothingness that is part of modern relational databases was introduced by Codd's original proposal in 1970 for the relational model (Codd, 1970), which is nearly contemporaneous with Hoare's invention of the null reference. The relational symbol for the lack of a value in the field of a record is NULL.

Ever since Codd introduced NULL, it was controversial. The main problem is that the concept is underspecified, resulting in inconsistent implementations. Even to the extent that it is specified, it is inconsistent, and explanations for the inconsistencies are not convincing. Many researchers have published, and continue to publish, proposals for correcting the difficulties with NULL. For example, Date and Darwen advocated in their Third Manifesto that the concept be eliminated entirely from relational database systems, replacing it with the programming language concept (Darwen and Date, 2006).

So what is the database NULL? As we mentioned in the previous section above, a database NULL is the absence of a value, while for most programming languages, null is one more value, albeit a special one. This has a number of consequences. The first is that comparing a database NULL with another database NULL, even the same database NULL, results in NULL, not TRUE. This makes sense because a database NULL means that there is no value, so comparisons are meaningless. One consequence is that one cannot use ordinary Boolean logic. The result of a logical expression now has three possibilities rather than just two. In addition to TRUE and FALSE, one must now admit the possibility of NULL. The resulting logic is an example of a three-valued logic, while ordinary Boolean logic is a two-valued logic. Strictly speaking the third "value" of the SQL

three-valued logic is not a value but rather the absence of a value, but for the purposes of evaluating a logical expression it is convenient to regard the third possibility as being another value.

One of the consequences of the three-valued logic is that one cannot test that the field of a record has a value by comparing the field with NULL. Doing so will always produce NULL whether the field has a value or not. Relational database systems have a special syntax for this purpose. One must say "field IS NULL" or "field IS NOT NULL."

A more significant problem with the database NULL is that it has led to confusion. For example, the term "null value" is an oxymoron for databases. Yet the phrase is used throughout database textbooks and articles on the Web such as in Wikipedia. Textbook writers even go so far as to make statements such as, "Nulls behave strangely in queries." The strangeness is entirely due to mistakenly believing that the database NULL is a value. When one properly understands that the database NULL is the absence of a value, then the strangeness disappears. Another example is, "A weird aspect of nulls is that they are neither equal nor unequal to each other." Once again, the weirdness is due to a misunderstanding rather than an intrinsic property of databases.

Unfortunately, while SQL generally regards NULL as being the absence of a value, it is not entirely consistent. For example, when records are sorted, NULLs in different records are regarded as being equal.

Even Codd himself admitted that his concept of NULL was flawed, and he attempted to fix it by introducing two different concepts of NULL (Codd, 1990). However, this suggestion was never implemented, largely because of the enormous increase in complexity that it would entail. For example, logic would have to increase from being three-valued to being four-valued. Furthermore, Codd's suggestion would not have fixed the various inconsistencies involving NULL. As we discuss below, it has the further disadvantage that there are many more than just two NULL concepts, and even the two NULL concepts suggested by Codd have their own subclassifications. Attempting to fix the problem by making more distinctions is clearly not a solution.

It is interesting to note that the inventors of their respective notions

of nothingness both regarded their inventions as a mistake.

Other Data Languages

More recently, data representations and data languages other than the relational databases have become popular. Prominent examples include the RDF and OWL representation languages as well as query languages such as SPARQL. For the rest of this section, we assume some familiarity with RDF and OWL.

A fundamental distinction between relational databases and RDF/OWL is whether they are open or closed. A logical system satisfies a closed world assumption (CWA) if some statements that are not currently known to be true can be inferred to be false. By contrast if a logical system satisfies the open world assumption (OWA), then lack of knowledge need not imply falsity. There are many kinds of CWA logical systems as well as many kinds of OWA logical systems, as well as logical systems that have aspects of both assumptions. The advantage of CWA is that query processing is much more efficient compared with OWA. However, CWA requires that a situation be completely known to the database. This is acceptable for a business database but is not reasonable for general knowledge queries.

A relational database is primarily CWA, but NULL can be regarded as an OWA feature in the midst of a system that is otherwise CWA. Most programming languages may be regarded as being CWA. Unlike relational databases, programming language nulls are values and so cannot be regarded as being an OWA feature. RDF, OWL and their query languages are defined to be OWA logical systems, although some aspects of CWA are sometimes employed for the sake of efficiency.

To give some idea of the consequences of OWA compared with CWA, consider a university database where students may have a major in a department, but the university has decided that no student can major in more than one department. Suppose that in spite of this rule a student has declared that they are majoring in English and Computer Science. In a CWA system, this would immediately result in an error because the major constraint is not satisfied. In an OWA system, this might not be an error. Instead, the OWA system would proceed to infer that English and Computer Science are the same department. A cascade of other inferences

might then ensue. For example, if a department is allowed to have no more than one head, then the heads of English and Computer Science would be inferred to be the same. Presumably, the cascade of inferences would eventually result in an unacceptable inconsistency, but by then the original reason for it might not be easy to trace.

In RDF and OWL the most common way to specify that a resource does not have any value for a property is to have no statement that has the resource as its subject and the property as its predicate. This is essentially the same as a database record having a NULL field. Indeed, some relational database implementations represent the fact that a field is NULL by omitting the field in the record.

Another way that one can, in principle, specify that a resource does not have a value for a property is to specify that its value is an empty list. The advantage of the list construct is that it allows one to specify exactly all of the values in a collection. In particular, it could be used to specify exactly all of the values that a resource has for a property. Accordingly, a list can be regarded as a CWA feature in the midst of a system that is otherwise OWA.

Some Interpretations of Nothingness

Whether in programming languages or data languages, NULL can arise for a great variety of reasons, and attempting to capture even a few of these in a language would be very difficult if not impossible. Here is a partial list of the reasons why NULL can occur.

1. The field has a value but it isn't known. There are a number of variations on this. Here are a few of them:

   (a) The value does exist somewhere, but the database does not have it.

   (b) The value has not yet been determined, but it presumably will be determined at some point. For example, a customer has yet to make a decision about a purchase.

   (c) The value does exist in principle, and it might someday be determined, but there is no guarantee that it will ever be determined.

For example, a field might have value 1 if P=NP and 0 if P≠NP.[2]

All of these may be regarded as examples of open world assumptions in the midst of a system that is otherwise closed. They have the common feature that the field could, in principle, be specified with a value at some time in the future.

2. The field has no value because it isn't applicable. Unlike the example above, there is no possibility that such a field could ever have a value. This can happen for a variety of reasons.

   (a) The field is not allowed to have a value. For example, the root node of a tree has no parent node. The fact that the field is NULL in this case is not due to missing information. As another example, in programming languages, a null reference is sometimes employed to mark the end of a sequence of objects.

   (b) The field was produced during an operation such as an arithmetic or data manipulation operation.

3. The field has a value, but it is not within the domain. For example, a form requesting an ethnic group does not include the one to which a person most closely identifies. The NULL in this case represents "none of the above" or "other." On such a form, NULL could also mean that one does not wish to answer the question, one does not know the answer, one neglected to answer the question, or many other possible explanations.

4. A field value could have been inferred but was not inferred, because of an overriding requirement. The example given here requires some knowledge of SQL. Specifically, it requires the outer join operation. The outer join mandates that the additional (padded) columns of an unmatched record be NULL even when one could infer values for the columns. Consider the following SQL statements:

[2]The P=NP problem is an unsolved problem in computer science. It asks whether every problem whose solution can be quickly verified can also be quickly solved, where "quickly" means "achieved in a period of time that is a polynomial in the size of the problem."

```
create table A(id int primary key, foo int default 0);
create table B(id int primary key, bar int default 0);
insert into A values(1, 10);
select * from A left join B on (A.id = B.id);
```

The result of the SQL statements above is the following:

id	foo	id	bar
1	10	NULL	NULL

(a) The second id field in the result must be NULL because the join condition did not hold in this case. It means that the A record was not matched with any B record. This represents a situation where a field cannot have a value.

(b) The bar field in the result could be inferred to be 0 because of the default value for this column. In spite of this, the bar field in the result is NULL. One could argue that this is also a situation where a field cannot have a value. In this case, there is no B record being joined, so it is not meaningful for it to have any value, even the default value. On the other hand, the usual purpose of a default value is that it is the value one should use in the absence of any other value being available. That certainly is true in this case, so there is also a good argument for using the default value rather than NULL.

5. The field value cannot be determined due to an exception. For example, dividing 1 by 0 is clearly an exception since division by zero is never defined for any number.

There are many other possibilities for the interpretation of nothingness. During query processing, computations use a three-valued logic which will introduce NULLs that have interpretations that are subtly different from the interpretations of the NULLs that occurred in the expressions being computed. For example, if NULL means that the value has not yet been specified, then the result of join processing should be that there could be a match but it is not yet known. By contrast, if NULL means "none

of the above," then one could argue that two NULLs should match one another.

One reason why the database notion of NULL is still useful in spite of its semantic shortcomings has to do with query optimization. Unlike programming languages which have relatively limited opportunities for optimization, there are substantial opportunities for optimizing queries. For example, a query that filters the results as the last step could be optimized so that the filter operation is performed much earlier in query processing, well before operations such as joins are performed. While this has important consequences for performance, it also means that standard programming language features like throwing and catching exceptions are no longer possible. By rearranging the processing steps the way databases do, one could be in a situation where the exception is caught at a different place in the computation than one would expect. In other words, the throw-catch notion is only meaningful for a specific execution plan, so throwing and catching exceptions within the processing of a query would mandate a specific execution plan and eliminate many possibilities for query optimization.

Still another kind of optimization is parallel processing (via multiple threads) to obtain query results. When an exception is thrown in a parallel context, it is unclear how the exception should be handled. Aside from the issue of where the exception should be caught, there is also the issue of whether other threads should be affected by the exception. The use of NULL and NULL propagation rules allows query processing to be optimized and parallelized, but at the cost of a notion of NULL that differs from the one for programming languages as well as a logic that is three-valued rather than two-valued.

Conclusion

The notion of nothingness has a long and complex history. It is both a philosophical issue that continues to be argued and a pragmatic issue that continues to have an impact on modern systems. We have given some idea of the difficulties faced by anyone attempting to specify the semantics of nothingness in a way that satisfies all relevant interpretations and use cases, while being compatible with the requirements for system performance.

References

Cartlidge, J. (n.d.), A summary of Heidegger's "What is Metaphysics?" https://bit.ly/3FwAPzA

Codd, E.F. (1970). "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM* **13**(6): 377-387.

Codd, E.F. (1990), *The Relational Model for Database Management Version 2*, Addison-Wesley, 1990 (ISBN 0201141922).

Darwen, H. and Date, C.J. (2006), *Databases, Types, and The Relational Model: The Third Manifesto*, 3^{rd} edition, Addison-Wesley. (ISBN: 0-321-39942-0).

Gheverghese, J. (2011). *The Crest of the Peacock: Non-European Roots of Mathematics (Third ed.)*. Princeton UP. p. 86. ISBN 978-0-691-13526-7.

Goth, M. (2013). Translation of Heidegger's "What is Metaphysics?" and other essays. https://bit.ly/3MeNB9d

Graham, D.W. (2008). "Leucippus' Atomism". In Curd, Patricia; Graham, Daniel W. (eds.). *The Oxford Handbook of Presocratic Philosophy*. Oxford University Press. ISBN 978-0-19-514687-5.

Hegel, G.W.F. (1812-1816), *Wissenschaft der Logik*. https://bit.ly/47t41mt

Heidegger, M. (1927), *Sein und Zeit (Being and Time)*.

Heidegger, M. (1929) "What is Metaphysics?"

Hoare, T. (2009). "Null References: The Billion Dollar Mistake." InfoQ.com.

IEEE 754 (2019). IEEE Standard for Floating-Point Arithmetic. IEEE STD 754-2019. IEEE. pp. 1–84. IEEE Computer Society (2019-07-22). https://bit.ly/47ckjAK

Inwood, M. (1999). Does the Nothing Noth? Royal Institute of Philosophy Supplements, **44**: 271-290. http://bit.ly/3yLv6SS

Menninger, K. (1992). *Number words and number symbols: a cultural history of numbers*. Courier Dover Publications. pp. 399–404. ISBN 978-0-486-27096-8.

O'Connor, J. and Robertson, E. (2000). "Egyptian numerals". mathshistory.st-andrews.ac.uk. https://bit.ly/45N3JWz

Russell, B. (1995). *History of Western Philosophy*, Routledge ISBN 0-415-07854-7.

Sartre, J.-P. (1943), *L'Être et le Néant (Being and Nothingness)*. Trans. Hazel E. Barnes. New York: Washington Square Press, 1984.

Skordoulis, C.D., Koutalis, V. (2013). Tsaparlis, Georgios (ed.). *Concepts of Matter in Science Education*. Springer. ISBN 978-94-007-5914-5.

Wallin, N.-B. (2002). "The History of Zero." The Whitney and Betty Macmillan Center for International and Area Studies at Yale. https://bit.ly/40kL9UM

MSVIPER: Improved Policy Distillation for Reinforcement-Learning-Based Robot Navigation

Aaron M. Roth[1,2], Jing Liang[1], Ram Sriram[2],
Elham Tabassi[2], and Dinesh Manocha[1]
[1] University of Maryland [2]National Institute for Standards and Technology

Abstract

We present Multiple Scenario Verifiable Reinforcement Learning via Policy Extraction (MSVIPER), a new method for policy distillation to decision trees for improved robot navigation. MSVIPER learns an "expert" policy using any Reinforcement Learning (RL) technique involving learning a state-action mapping and then uses imitation learning to learn a decision-tree policy from it. We demonstrate that MSVIPER results in efficient decision trees and can accurately mimic the behavior of the expert policy. Moreover, we present efficient policy distillation and tree-modification techniques that take advantage of the decision tree structure to allow improvements to a policy without retraining. We use our approach to improve the performance of RL-based robot navigation algorithms for indoor and outdoor scenes. We demonstrate the benefits in terms of reduced freezing and oscillation behaviors (by up to 95% reduction) for mobile robots navigating among dynamic obstacles and reduced vibrations and oscillation (by up to 17%) for outdoor robot navigation on complex, uneven terrains.

I. Introduction

LEARNING METHODS ARE INCREASINGLY BEING USED for robot navigation in indoor and outdoor scenes. These include deep reinforcement learning (DRL) methods [11, 39, 20, 23, 31, 22, 29] and learning from demonstration [2, 34]. These methods have been evaluated in real-world scenarios, including obstacle avoidance, dynamic scenes, and uneven terrains. In these applications, the performance of the navigation methods varies based on the underlying learned policy. It turns out that the learned policy may have errors or be otherwise sub-optimal. Additionally, the learned policy is often represented in the form of a neural net, which can be rather difficult to analyze.

In this paper, we are mainly interested in developing approaches to Reinforcement Learning that use a decision-tree (DT) policy. A decision tree can be more easily analyzed and modified, and the cause-and-effect of

state and action is apparent. Learning a successful tree policy is typically more difficult than learning with a neural net. Moreover, these issues are compounded when one attempts to produce a policy for use beyond toy problems and proof-of-concepts [17, 12, 28]. We further investigate using the tree policy to enable modifying and improving upon a policy when the original learning pipeline becomes unavailable. In the real world, there will be situations where we want to take a learned policy for an autonomous robot, and analyze its shortcomings (*including* shortcomings defined or noticed after initial training!) and then fix those shortcomings. Retraining will not always be possible, nor will the original dataset always be available.

Figure 1: MSVIPER for outdoor (left) and indoor (right) navigation

We introduce a novel method, Multiple Scenario Verifiable Reinforcement Learning via Policy Extraction (MSVIPER), and use it for improved robot navigation in complex indoor and outdoor scenes. In Figure 1 the left image is from the uneven outdoor terrain environment navigation. Our policy distillation and tree modification reduce oscillation and vibrations in the resulting trajectory. The right image of Figure 1 shows a test of the real-world tree policy for indoor obstacle avoidance, where MSVIPER results in smoother trajectories and overcomes freezing problems. Our formulation extends VIPER [4] and learns an "expert" policy using a neural net (such as PPO [33]) before using imitation learning to fit a DT to replicate the expert policy. MSVIPER compensates for some of VIPER's limitations and also adds new capabilities. There has been limited work on applying VIPER to more complex tasks and almost no work on applying it to any real-world applications. Even for the tasks to which it has been applied, there is no recourse for addressing any errors or issues discovered.

We use MSVIPER to improve the performance of reinforcement-

learning-based robot navigation methods. Instead of using different heuristics or trying different parameters and repeating the time-consuming training step, we integrate expert policies, policy extraction, and decision trees. Our *policy distillation* approach directly analyzes and performs tree modification operations on the DTs. Some of the novel contributions of our work include:

1. We present MSVIPER to improve imitation learning of DTs for indoor and outdoor robot navigation policies. MSVIPER samples from trajectories in multiple environmental scenarios. MSVIPER also results in smaller DTs, which are easier to understand and analyze [26]. We also demonstrate the superior sampling complexity of MSVIPER.

2. We describe techniques to modify and improve an initial tree policy *without retraining*, which could not be performed using a standard neural net policy. We use these improved tree policies to generate smooth paths for collision-free robot navigation, to fix freezing problems in crowded indoor scenarios, and to reduce vibrations and oscillations on complex outdoor terrains.

3. We highlight the performance of improved policies on indoor and outdoor navigation scenarios. The indoor navigation is performed using a Jackal robot and Turtlebot among moving pedestrians and we observe up to 95% improvement in terms of reduction in freezing and oscillation behaviors. The outdoor navigation is performed using a Clearpath Husky and Jackal on uneven terrains with obstacles. We observe up to 17% improvement in terms of reduced vibrations and oscillations.

II. Background and Related Work

There is considerable work on using machine learning methods for robot navigation, interpretation of networks [15] and decision trees [28]. A soft decision tree can be distilled from a deep neural network [17] or learned directly via a Policy Tree [18]. A tree can be learned in an additive manner using a reinforcement-learning-style approach [25] and Conservative Q-Improvement [30].

Policy distillation involves transformation of a policy from one format to another while keeping the essential input-output as similar as possible. This could be transforming a neural net or Markov Decision Process (MDP) into a smaller network or MDP [14], or into a different form entirely, such as a saliency map [38] or tree [21, 4]. For policy distillation, our approach is based on VIPER [4] because it is flexible in terms of being able to use any method to learn the expert policy. In practice, VIPER's usage has been limited to proof-of-concept problems such as Cartpole [6], Pong [4] and other simulations such as CARLA [10].

Autonomous systems need the ability to deal with uncertain and unforeseen scenarios. Some existing systems reduce risk or correct errors by gathering information from a human through methods such as learning from demonstration [7, 9, 8]. Human feedback can also incorporate missing state features into state representation [3]. One approach asks a user to label states of interest as a means of gaining information during training [35]. The above-mentioned approaches incorporate human expertise while remaining black boxes. They typically require retraining, which can be time consuming. In our approach, we can evaluate the effectiveness and efficiency of the learned policy and directly improve without retraining.

III. Problem Formulation and Notation

As is typical in approaches that make use of reinforcement learning methods, we model the underlying task in the application as a Markov Decision Process (MDP), which could be represented by a 5-tuple (S, A, P, R, γ). S is state space, A is action space, R is rewards, and P is state transition dynamics: $S \times A \rightarrow S$, and $\gamma \in (0, 1]$ is the discount factor. Distribution of initial state is represented as ρ_0. We want to generate an optimal policy π that maximizes discounted reward function:

$$\eta(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r(s_t, a_t) \right].$$

where T is the time horizon. The policy is represented by networks with parameter θ_π.

In the Deep RL learning step, we want the robot to learn to accomplish one of the three given tasks, involving navigating to a static or dynamic goal position, avoiding collisions with static or dynamic obstacles,

and achieving a smooth trajectory that avoids vibrations and oscillations.

This policy is black-box, it may be suboptimal, and it may have errors a certain percentage of the time. We seek to address such issues during the distillation and modification procedures described in Section IV.

We assume the researcher or developer working with the model has access to the model itself in the form of a black box, but does not have the capability to retrain it (simulating technological, temporal, organizational, legal, or other constraints that arise in the real world).

The notation we use is shown in Table 1.

IV. Our Approach: MSVIPER

We present a pipeline in Figure 2, the end-result of which is a modifiable decision tree policy for robot navigation. In this figure we highlight three parts of MSVIPER in the top row. First, we train an expert policy with different scenarios. Second, we generate trajectories using the expert policy. Third, we use the trajectories to train a decision tree. Next, we use our tree modification techniques to correct errors and improve navigation performance without retraining, as shown the bottom row of Figure 2. Section IV-A discusses use of deep reinforcement learning to create an initial learned navigation policy. Section IV-B describes policy distillation into a decision tree using MSVIPER. Section IV-C describes our tree modification procedure to improve the policy for robot navigation. We highlight its benefits in terms of addressing the issues related to freezing, oscillation, and vibrations in a robot's trajectory.

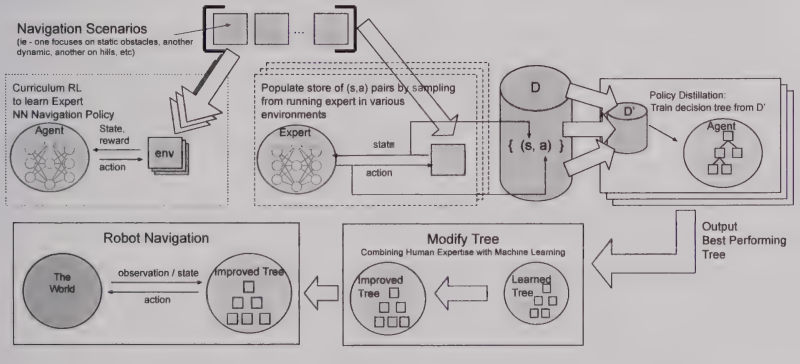


Figure 2: Overview of our Policy Distillation and Tree Modification

$s \in S$	a state s is an array representing the state of the world and the robot in it
$a \in A$	an action a is a single discrete action in the set of possible actions A
P	state transition dynamics
r, R	reward (for a single (s, a) pair or in general, respectively)
γ	future discount factor
π	a policy (whether neural net or tree based)
π^*	an expert policy (neural net)
$\hat{\pi}$	a decision tree policy
l_t	length of trajectory to use when generating state-action pairs
M	number of trajectories used while generating state-action pairs
N	number of VIPER iterations used to run per scenario
E	an ordered list of environments/scenarios (repetition is allowed)
n_{cv}	the number of cross-validation trials used to compare the candidate final policies $\hat{\pi}$
D, D_i, D'	unordered sets of state action pairs ($\{(s,a)\}$)
n_s	number of samples to use when generating D' from D
e_O	Objective Efficiency, a measure of how efficiently a tree modification improved a policy; is higher when the target metric has higher improvement per node-modified
e_R	Relative Efficiency, a measure of how much a tree modification improved a policy; is higher when the target metric has higher improvement per percentage-of-nodes-modified
M_1, M_2	values of a target metric before and after tree modification
N_+	Number of nodes added, removed, or modified in a tree modification procedure
N_1	Number of nodes in $\hat{\pi}$ before modification

Table 1: Symbols and notation used in the paper

A. Reinforcement Learning for Initial Navigation Policy

We solve the Deep RL task as formulated in Section III. We trained using a standard Deep RL algorithm such as the Proximal Policy Optimization (PPO) [33] algorithm. One important detail to note is that we trained our agent in a succession of increasingly difficult environment stages and not in a single stage. Many methods use Curriculum Learning (CL) [5] to learn good navigation behavior. In CL, instead of a single environment, the agent is trained in a succession of increasingly difficult environment stages. In CL, the policy trained by each scenario is reused as an initialization of the next scenario of training. In particular, the final policy from an earlier scenario becomes the starting policy for a later scenario. We used CL in the Deep RL stage, creating multiple scenarios of increasing difficulty in each case and use them to train a neural net format policy.

B. Policy Distillation: Conversion to Decision Tree Policy

We perform policy distillation on the initial policy to create a DT policy. First, we generate trajectories, which are sequences of state-action pairs $(\{(s_0, a_0), ..., (s_i, a_i)\}, s \in S, a \in A)$ in a manner that is more sample efficient for some environments than previous approaches. Prior methods (VIPER [4]) would sample trajectories generated by applying the policy in a single simulated environment, and use those trajectories to train a decision tree. We refer to it as single source VIPER (SSVIPER). Instead, we sample the trajectories generated by applying the policy in multiple simulated environments of varying complexities or types. We try to ensure a large variety in terms of generating indoor and outdoor simulation environments. This variety ensures that the DTs that result from learning based on the sampled pairs will be more robust to different scenarios that might be encountered in the real world scenarios.

Algorithm 1: Multi-Scenario VIPER:
Our improved algorithm for learning a decision tree policy through imitation learning of an expert policy across multiple scenarios

MSVIPER($(S, A, P, R), \pi^*, M, N, E, l_t, n_{cv}, n_s$):
$D \leftarrow \varnothing$;
for *scenario* $e \in E$ **do**
  for $i = 1$ *to* N **do**
    $s_0 \leftarrow$ initial state after resetting e;
    $D_i \leftarrow \{(s_j, \pi^*(s_j)) \sim d^{s_{j-1}}\}$ (sample M trajectories of length l_t, by running policy π^* in environment scenario e);
    $D \leftarrow D \cup D_i$;
    $D' \leftarrow \{(s,a)\ \{(s,a)\} \in D\}$ (sample a dataset, retrieve a sample size of n_s pairs);
    Train decision tree $\hat{\pi}_i$ using D' as dataset;
  end
end
Return best policy $\hat{\pi}_i \in \{\hat{\pi}_1$ to $\hat{\pi}_N\}$ using n_{cv} trials to check each of them.

In MSVIPER (Algorithm 1), for each scenario $e \in E$, we use this procedure: For a sequence of N iterations, we sample trajectories of length M. All such pairs are put into a set. Pairs are randomly drawn from that set and used as a dataset to train a DT in a supervised learning manner using CART [19]. The overall dataset from which pairs are drawn is maintained between the stages. The resulting N decision trees are evaluated in simulation for n_{cv} trials, and the best-performing (highest reward) tree is the output of the method.

C. Tree Modification: Policy Optimization & No Retraining

We use our tree-based policy for analysis, verification, and modification. We modify the tree to improve the navigation characteristics of the policy without retraining.

A key issue in terms of tree modification is to change as few nodes as possible. Details of our novel algorithms for different types of modifications to improve navigation can be found in the Appendix, and are summarized below. Tree modifications can include changing the action on a leaf node,

changing the feature a node splits on or the threshold at which it splits, removing nodes, or adding new nodes.

(1) Indoor Navigation: Tree Modification

We address two main issues with respect to learning-based methods used for indoor navigation: freezing robot problems and oscillations. A common issue in terms of robot navigation in indoor scenes among dynamic obstacles is the "Freezing Robot Problem" [37, 36, 32, 16]. It corresponds to configurations, where the robot is presented with a set of obstacles and chooses to remain immobile to avoid any collision. We modify the policy to reduce such instances. Our freezing identification procedure identifies nodes in a decision tree that cause (or could potentially cause) the freezing behavior and change the attributes of the identified nodes. The algorithm analyzes each leaf node in the tree. If the node's action is the Stop action, and if the occupancy grid described by the state bounds of that node indicate stationary obstacles (within a configurable tolerance for minimal movement), then the node is considered to be a problematic node and is added to the list of potential-freezing nodes. Each potential-freezing node is adjusted, given a rotation action.

We also modify the tree to generate smoother trajectories and reduce the oscillations. The oscillatory behavior can occur when a robot moves around the obstacles. Oscillations can be defined in two ways: first as a percentage as in $C_{osc} = (c(a_i)/T)\forall i = \{0, ...T\}$, where T is a number of timesteps and $c(a_i) = 1$ if the action a_i is part of a sequence of actions of certain length that alternate between left-right rotation angular velocity, $c(a_i) = 0$ otherwise; and secondly, as $C_{\text{osc}} = \frac{1}{N}\sum_{i=0}^{N} abs(\omega_t - \omega_{t-1})$, where ω is the angular velocity of the robot and the equation indicates the average changes in the angular velocities. We use the simulations to record such behaviors and use that to identify the corresponding potentially oscillating nodes in the tree. We modify the tree by modifying existing nodes or adding nodes (depending on calculations of how much of the problem node's subspace is or is not affected by the same problem), replacing existing actions with replacement actions that have lower linear and angular velocities.

(2) Outdoor Navigation: Tree Modification

In outdoor navigation, a robot needs to deal with complex, uneven terrains with obstacles. This tends to cause jerky motion with bounces and

high vibrations. We define robot vibration as:

$$V_b = \sum_{t'=t-3}^{t} \gamma^{t-t'}(|\omega_r| + |\omega_p|) = \sum_{t'=t-3}^{t} \gamma^{t-t'} w_{t'} \tag{1}$$

where $\omega = |\omega_r| + |\omega_p|$. V_b is the vibration of the robot, where γ is the discount factor and ω_r and ω_p are the angular velocities in the roll and pitch directions. We analyze the robot's angular velocity and identify the corresponding potentially vibrating nodes. We look at nodes that split on angular velocity as well as nodes that exist inside or intersect the "vibration space" (the closed hyperdimensional object described by V_b; this is a hyperrectangle and convex hull). Our formulation is able to treat the vibration threshold as a hyperparameter and tune it to decrease the vibration in the learned policy. We can modify the tree by adjusting the threshold of certain nodes or by changing the action (and thus motion primitive) to reduce linear and/or angular velocity for that node. We are also able to modify the tree to reduce the oscillations, similarly to indoor navigation.

D. Analysis: Upper Bound Divergence

Compared to SSVIPER, MSVIPER makes smaller and more efficient DTs.

Theorem 1 *For MSVIPER, the loss function is :*

$$\tilde{l}_t(s,\pi) = V_t^{(\pi^*)}(s) - \min_{a \in A} \hat{Q}_t^{(\pi^*)}(s,a), \tag{2}$$

where

$$\hat{Q}^{\pi^*}(s,a) = \mathbb{E}_\pi \left[\sum_{e \in E} \left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}^e | S_t = s, A_t = a \right) \cdot w_e \right] \tag{3}$$

$\hat{Q}^{\pi^*}(s,a)$ is the quality function, $V_t^{(\pi^*)}(s)$ is the value function of the policy, R_i^e is reward at timestep i in environment e, and w_e weights the contribution of each environment. In order to guarantee improved performance, we need to decrease the upper bound on $\tilde{l}_t(s,\pi)$.

Lemma 1 *for a given π^*, sampling multiple scenarios:*

$$J(\hat{\pi}) \leq J(\pi^*) + T\epsilon_N + \tilde{O}(1) \tag{4}$$

where $J(\hat{\pi}) = -V(\hat{\pi})(\mathbf{s})$ be the cost-to-goal from the state $\mathbf{s}$. The ϵ_N is the training loss defined in [4]. T and N are time steps and total training iteration numbers, respectively. $\tilde{O}(1)$ is a constant function. Equation (4) holds true.

The proof of the theorem is in Appendix C.

V. Results

A. Environments

We tested the performance of MSVIPER in three environments: i) an indoor environment with obstacles for collision-free navigation; ii) an indoor warehouse environment where a robot must find and follow a human, and iii) an outdoor environment with uneven terrain a robot must traverse.

In the indoor Obstacle Avoidance environment, the robot starts in a random location and must navigate around static and dynamic obstacles to a random goal location. We desire that the policy used for interpretation should perform well in terms of avoiding the pedestrians and obstacles.

In the indoor Find and Follow environment, the robot spawns in a complex multi-room environment with obstacles, a warehouse full of haphazardly arranged boxes. It spawns in a room. It must exit the room and find and follow a human in a white shirt around a warehouse. There are three stages in this task: i) learning the exit the room, ii) learning to exit the room and finding the human, iii) following the human as they move.

The outdoor navigation environment has a starting location and goal location and uneven terrain between them. The robot must navigate over the terrain safely and efficiently. There are no obstacles in this environment but the challenge is optimally navigate over inclined or uneven outdoor terrain.

Environment	Policy Type	Final Avg Reward/ts (test)	Size (# Nodes)	Depth	Runtime(s)
Obstacle Avoidance	Expert NN	0.29	n/a	n/a	7.81
Obstacle Avoidance	SS VIPER	0.23	2651	25	8.84
Obstacle Avoidance	MSVIPER	**0.30**	411	16	7.59
Find & Follow	Expert NN	0.464	n/a	n/a	90.3726
Find & Follow	SS VIPER	0.097	2073	22	51.00
Find & Follow	MSVIPER	**0.19**	**1681**	20	34.02

Table 2: Improved Navigation using MSVIPER

B. MSVIPER: Improved Navigation

MSVIPER achieves a higher average reward, smaller trees, and faster runtime than SS VIPER. As shown in Table 2 we compared the performances by expert policies in simulation using MSVIPER and SSVIPER. MSVIPER generated DTs for the indoor navigation obstacle avoidance environment had a simulated reward near the expert policy level. Moreover, it has less nodes (smaller size) than the SSVIPER policy. MSVIPER exhibits better performance than SSVIPER. In the obstacle avoidance environment, The robot trained by MSVIPER has a better chance of going to the goal without any collisions. MSVIPER results in a smaller tree than VIPER, making it tree modification easier. In the Find and Follow scenario, MSVIPER results in smaller DT and improves the final average reward compared to SSVIPER. The superior reward for MSVIPER also indicates shorter navigation time for the robot. In all indoor environments, MSVIPER takes advantage of the curriculum stages to increase sample diversity and cover more critical states. It also reduces the chances of overfitting.

Figure 3 shows the sizes of trees as sample complexity changes. More samples result creation of trees that more closely mimic the expert policy. MSVIPER results in a smaller size tree than VIPER and that makes it easier in terms of tree modification.

Indoor Navigation Tree Policy	n_s	N_1	N_2	N_+	Freezing %:	(initial)	(modified)	e_O	e_R
Policy with Freezing Error	5000	319	319	12		100	5	0.079	25.25
					Oscillation %	(initial)	(modified)		
Policy with Excessive Oscillation	30,000	1429	1429	11		96	6	0.085	121.7

Table 3: Tree Modification for Indoor Robot Navigation

C. Indoor Navigation Results

For indoor navigation, we validated our learned decision tree policy's ability to handle navigation in a real-world environment using two robot platforms, Clearpath Jackal [1] and Turtlebot. We tested the policy on both robots in lab environments with various furniture and moving humans serving as obstacles. We observe significant reductions in the freezing behaviors with dynamic obstacles and fewer oscillations using MSVIPER and tree modifications (see Table 3). We observe significant improvements, by $90-95\%$, by comparing the paths computed by the initial navigation policy verses the modified policy.

D. Outdoor Navigation Results

We tested our trees and also the expert policy in the real-world outdoor environment with uneven terrain using a Clearpath Husky robot. The trained trees have very similar performances to the expert policy. After the tree modification the vibration and oscillation of the robot both reduce and the robot runs much more stably and smoothly. We highlight the benefits of policy distillation and tree modification in terms of reduced vibrations and oscillations on uneven outdoor terrains by 17% (see Table 4).

E. No-Retraining Tree Modification: Efficiency Metrics

First, for each navigation behavior (freezing, oscillation, vibration) we create a domain-specific procedure that can detect whether a given tree policy contains nodes that could results in those behaviors, and which detects those nodes. We are able to perform this analysis with a DT, but would be much harder with a neural net structure. Second, given the above, we can modify the policy to mitigate or fix the issue. We evaluate the policies according to navigation metrics before and after the

Outdoor Navigation Tree Policy	n_s	N_1	N_2	N_+	Vibration:	(initial)	(modified)	e_O	e_R
Policy with Excessive Vibration	10,000	17	17	1		0.53	0.48	-0.084	8.29
					Oscillation %	(initial)	(modified)		
Policy with Excessive Oscillation	10,000	17	17	2		0.092	0.076	0.089	1.51

Table 4: Tree Modification for Outdoor Robot Navigation

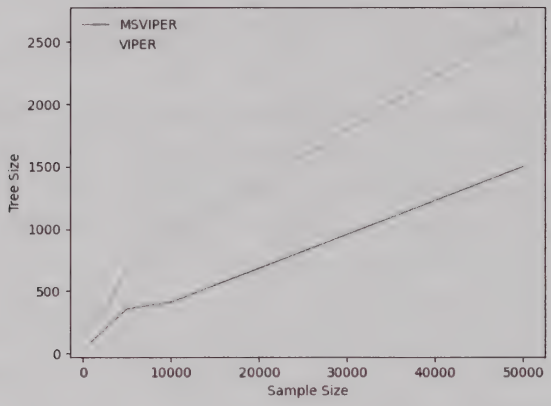


Figure 3: Size of Trees Generated for VIPER and MSVIPER. MSVIPER produces smaller trees for the same total number of samples, on average. Smaller trees are beneficial from an interpretability standpoint [26].

trees are modified, to determine whether there was improvement. We can also measure the improvement with tree modification efficiency metrics.

In particular, we evaluate the benefits of tree modification using two metrics. Table 3 and 4 show the **Objective Efficiency** e_O and the **Relative Efficiency** e_R of the modifications. Each metric is formulated with respect to a target metric M that the tree modification is attempting to accomplish. In our demonstrations, M is the percentage of trials where freezing occurs (for freezing), C_{osc} (for oscillation), and V_b (for vibration). If we need to make a larger change in the initial policy to achieve the desired change in the target metric, it results in lower efficiency. These metrics are computed as:

$$e_O = \frac{|M_2 - M_1|/M_1}{N_+}, \quad e_R = \frac{|M_2 - M_1|/M_1}{N_+/N_1}, \tag{5}$$

where M_1, M_2 are the values of the target metric (either %-of-trials-where-freezing-occurs or %-of-trials-where-oscillation-occurs) before and after tree modification, respectively, N_1 is the number of nodes before modification, and N_+ is the number of nodes modified (changed or added). As shown, e_O is higher when the target metric has higher improvement per node-modified, and e_R is higher when the target metric has higher improvement per percentage-of-nodes-modified.

For example, in the best-case freezing fix for MSVIPER, running

Figure 4: We have evaluated MSVIPER on a Jackal Robot and a Turtlebot navigating in real-world scenarios with pedestrians and obstacles. The arrow indicates the path taken by the robot as a result of the policy, showing it successfully navigating around the obstacles. We observe fewer freezing and oscillatory behaviors using MSVIPER.

the freezing-detection algorithm revealed that 30 out of 199 nodes of the tree were configured such that they could lead to a freezing issue in the initial learned policy.

We analyzed many aspects of computing efficient DTs and tree modifications. One additional benefit of the tree modification metric, in addition to quantifying improvement, is higher efficiency of MSVIPER over SSVIPER. For oscillation reduction in indoor navigation, the tree improved using our MSVIPER method has a higher objective and relative efficiency than that improved using SSVIPER (the MSVIPER e_O and e_R are shown in Table 3, VIPER has $e_0 = 0.032, e_R = 32.9$, both lower that MSVIPER's). Since the initial MSVIPER policy closely mimics the expert policy, only 11 nodes require modification to fix the oscillation, as compared to 31. To improve the freezing behavior, the most-improved SSVIPER policy had an n_s of size 5K (shown in the Table) compared to the best performing MSVIPER at $n_s =$ of size 1K. Freezing behavior was not present in *all* initial policies. MSVIPER at a 1K sample size resulted in no freezing behavior. In other words, MSVIPER resulted in a policy with fewer errors from fewer samples, and there were fewer errors to correct in the best-case policy (in the case of freezing). This highlight many benefits of MSVIPER in terms of efficient DTs and policy distillation.

F. Discussion of Implications

We have shown that policies from MSVIPER can be improved by tree modification without retraining. This is of critical importance in real-life applications. In some applications, we are often interested in the initial training process. In scenarios where the users will train and use RL models, however, there is great utility in a pipeline that allows for modifying a policy after-the-fact. This has applicability in situations where training is prohibitively expensive, for example, as well as situations where customization is useful. In particular, it allows improving and modifying policies by a software engineer or other developer without specific knowledge of reinforcement learning.

The examples involved here involved improving upon a particular metric, correcting an error, and even converting a failing policy into a successful one. One can imagine application developers creating solutions to error-prone or optimal learned policies and improving upon them further, delivering results beyond those that could be achieved by man or machine alone.

VI. Conclusion, Limitations, Future Work

We introduce MSVIPER, which learns a decision-tree-style policy for complex navigation tasks by imitating a neural-net style policy and enable post-training modifcation. We demonstrate the validity of approach in simulation environments and in real-world scenarios in the context of a robot navigating obstacles in a crowd, a robot following a person in a warehouse and robot navigation in complex outdoor terrains. We show examples of our policy structure allows it to be modified and improved without retraining.

Limitations exist: MSVIPER requires a discrete action space, a numerical-array-format state space, and hyperparameter tuning (its predecessors share these limitations). Our approach is best suited for scenarios where training can be expensive and it would be useful to reduce the overhead of retraining with different parameters.

Future work could involve using MSVIPER for applications of reinforcement learning beyond robot navigation. Additionally, our tree modification could be used on trees generated via other means.

References

[1] Jackal ugv - small weatherproof robot - clearpath. https://bit.ly/3ScfYsL. Accessed: 2021-08-17.

[2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] Connor Basich, Justin Svegliato, Kyle Hollins Wray, Stefan J Witwicki, and Shlomo Zilberstein. Improving competence for reliable autonomy. *arXiv preprint arXiv:2007.11740*, 2020.

[4] Osbert Bastani, Yewen Pu, and Armando Solar-Lezama. Verifiable reinforcement learning via policy extraction. In *Advances in Neural Information Processing Systems 31*, pages 2494–2504. Curran Associates, Inc., 2018.

[5] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[6] Surya Bhupatiraju, Kumar Krishna Agrawal, and Rishabh Singh. Towards mixed optimization for reinforcement learning with program synthesis. *arXiv preprint arXiv:1807.00403*, 2018.

[7] Daniel S Brown, Yuchen Cui, and Scott Niekum. Risk-aware active inverse reinforcement learning. In *Conference on Robot Learning*, pages 362–372. PMLR, 2018.

[8] Daniel S Brown and Scott Niekum. Toward probabilistic safety bounds for robot learning from demonstration. In *2017 AAAI Fall Symposium Series*, 2017.

[9] Peide Cai, Hengli Wang, Huaiyang Huang, Yuxuan Liu, and Ming Liu. Vision-based autonomous car racing using deep imitative reinforcement learning. *IEEE Robotics and Automation Letters*, 2021.

[10] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

[11] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017.

[12] Stephen P Curram and John Mingers. Neural networks, decision tree induction and discriminant analysis: An empirical comparison. *Journal of the Operational Research Society*, 45(4):440–450, 1994.

[13] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[14] Florent Delgrange, Ann Nowe, and Guillermo A Pérez. Distillation of rl policies with formal guarantees via variational abstraction of markov decision processes. *AAAI 2022*, 2022.

[15] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[16] Tingxiang Fan, Xinjing Cheng, Jia Pan, Pinxin Long, Wenxi Liu, Ruigang Yang, and Dinesh Manocha. Getting robots unfrozen and unlost in dense pedestrian crowds. *IEEE Robotics and Automation Letters*, 4(2):1178–1185, 2019.

[17] Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.

[18] Ujjwal Das Gupta, Erik Talvitie, and Michael Bowling. Policy tree: Adaptive representation for policy gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[19] Roger J Lewis. An introduction to classification and regression tree (cart) analysis. In *Annual meeting of the society for academic emergency medicine in San Francisco, California*, volume 14, 2000.

[20] Haoran Li, Qichao Zhang, and Dongbin Zhao. Deep reinforcement learning-based automatic exploration for navigation in unknown environment. *IEEE transactions on neural networks and learning systems*, 31(6):2064–2076, 2019.

[21] Zhao-Hua Li, Yang Yu, Yingfeng Chen, Ke Chen, Zhipeng Hu, and Changjie Fan. Neural-to-tree policy distillation with policy improvement criterion. *arXiv preprint arXiv:2108.06898*, 2021.

[22] Jing Liang, Utsav Patel, Adarsh Jagan Sathyamoorthy, and Dinesh Manocha. Crowd-steer: Realtime smooth and collision-free robot navigation in densely crowded scenarios trained using high-fidelity simulation. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4221–4228, 2021.

[23] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6252–6259. IEEE, 2018.

[24] Huansheng Ning, Rui Yin, Ata Ullah, and Feifei Shi. A survey on hybrid human-artificial intelligence for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[25] Larry D Pyeatt. Reinforcement learning with decision trees. In *Applied Informatics*, pages 26–31, 2003.

[26] J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[27] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[28] Aaron Roth. Structured representations for behaviors of autonomous robots. Master's thesis, Carnegie Mellon University, Pittsburgh, PA, July 2019.

[29] Aaron M Roth, Jing Liang, and Dinesh Manocha. Xai-n: sensor-based robot navigation using expert policies and decision trees. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[30] Aaron M Roth, Nicholay Topin, Pooyan Jamshidi, and Manuela Veloso. Conservative q-improvement: Reinforcement learning for an interpretable decision-tree policy. *arXiv preprint arXiv:1907.01180*, 2019.

[31] Adarsh Jagan Sathyamoorthy, Jing Liang, Utsav Patel, Tianrui Guan, Rohan Chandra, and Dinesh Manocha. Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11345–11352. IEEE, 2020.

[32] Adarsh Jagan Sathyamoorthy, Utsav Patel, Tianrui Guan, and Dinesh Manocha. Frozone: Freezing-free, pedestrian-friendly navigation in human crowds. *IEEE Robotics and Automation Letters*, 2020.

[33] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv e-prints*, page arXiv:1707.06347, Jul 2017.

[34] David Silver, J Andrew Bagnell, and Anthony Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. *The International Journal of Robotics Research*, 29(12):1565–1592, 2010.

[35] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

[36] Pete Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015.

[37] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 797–803. IEEE, 2010.

[38] Jinwei Xing, Takashi Nagata, Xinyun Zou, Emre Neftci, and Jeffrey L Krichmar. Policy distillation with selective input gradient regularization for efficient interpretability. *arXiv preprint arXiv:2205.08685*, 2022.

[39] Kai Zhu and Tao Zhang. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, 26(5):674–691, 2021.

Appendix

A. Tree Modification Algorithms

Unlike some other approaches that incorporate human input (such as [24]), in our approach we use human input to correct the errors while keeping the robot a fully autonomous entity at execution time. In this manner, we enable a synthesis of human and machine intelligence.

(1) Freezing

A common issue in robot learning for navigation is the "Freezing Robot Problem." [37, 36, 32, 16] Here the robot is presented with a set of obstacles and chooses to remain immobile. The purpose of choosing to remain motionless is to avoid crashing into an obstacle. However, it is also clearly undesirable since the robot has ceased to progress in the direction of the goal. One critical risk is that in the case of static obstacles causing this error, the failure case is one from which the robot cannot extricate itself. We used the modification procedures we outlined in [29] (Algorithms 1 and 2) to identify nodes that might potentially cause a freezing error and change the attribute of the identified node to reduce the risk of a freezing error.

(2) Oscillation

The next issue we address with regards to correcting the imperfect expert policies is the issue of oscillation. While traveling around obstacles, the robot will occasionally alternately rotate too far towards and away from the impediment. This behavior is inefficient and off-putting to a human observer. As before, we develop a procedure to mitigate the issue. As detailed in [29] Algorithms 3 and 4, we run the policy in simulation, record the robot's behavior, and note which parts of the decision-tree policy are activated when the undesirable behavior occurs (contributing to the behavior). We modify the tree by modifying existing nodes or adding nodes, replacing existing actions with replacement actions that have lower linear and angular velocities.

(3) Vibration

We introduce two tree modifications aimed at addressing excessive vibration. These can be used to address vibration when the learned policy

does not sufficiently reduce vibration below the threshold, or when one wants to modify the vibration threshold after the fact without retraining.

Algorithm 1: Detect Vibration Method 1

```
Detect Vibration Relevant Nodes( π†, V_E ):
N ← ∅;
for node n ∈ π† do
    if n is not a leaf node then
        if n[feature] ∈ V_E then
            Add n to N;
        end
    end
end
Return N
```

The first approach involves treating the threshold of nodes that split on a feature corresponding to angular velocity as a hyperparemeter, and modifying it to produce the desired result.

To identify relevant nodes, see the procedure in Algorithm 1. In our case, $\mathcal{V}_E = \{872, 873, 883, 884, 894, 895, 905, 906\}$, and represents the feature indices of features related to angular velocity (ω_r and ω_p at all timesteps present in the state). Any node in the tree that splits on one of these features is a relevant node.

In Algorithm 2, we show the procedure for adjusting the threshold at each of these nodes by an increment h. As noted, h is a hyperparameter that the human user can try out (since the tree modification is very fast) and tune to the desired level, measuring the vibration after each change.

Algorithm 2: Reduce Vibration Method 1

```
Modify Vibration-Relevant Nodes( π†, N, h ):
for node n ∈ N do
    n[threshold] ← n[threshold] + h;
end
Return the updated π†
```

There is a second Vibration reduction method that looks at the equation

$$V_b = \sum_{t'=t-3}^{t} \gamma^{t-t'}(|\omega_r| + |\omega_p|) \tag{1}$$

and notes that this also describes a closed hyperdimensional object. Specifically, for the four timesteps and two angular velocity values present above, it describes an eight-dimensional object. This object is a hyperrectangle and a convex hull. We call this the "Vibration Space."

Algorithm 3: Detect Vibration Method 2

Detect Vibration-Space-Intersecting Nodes($\pi^\dagger, \mathcal{V}_E, V_b, \gamma$):
$\mathcal{N}_V \leftarrow \varnothing$;
Let $\mathcal{H}_V$ be a set of points in a hyperspace with dimensions equal to the size of $\mathcal{V}_E$. These points form a convex hull describing the space taken by Equation 1, which we call the Vibration Space;
$\mathcal{C}_N \leftarrow$ a set of nodes initially containing only the root node;
while $\mathcal{C}_N \neq \varnothing$ **do**
    $n \leftarrow$ remove the last node from $\mathcal{C}_N$;
    Let $\mathcal{H}_N$ be a set of points forming a convex hull that is the boundaries of the abstract state space of n, considering only the dimensions indicated by $\mathcal{V}_E$;
    if *surfaces of* $\mathcal{H}_N$ *and* $\mathcal{H}_V$ *intersect* **then**
        Add n to $\mathcal{N}_V$;
    else
        Add both children of n, if any, to $\mathcal{C}_N$;
    end
end
Return $\mathcal{N}_V$

Consider Algorithm 3. The first two arguments to the procedure are the same as in Algorithm 1, and V_b, γ come from the vibration space equation. In a tree, the root node's abstract state space encompasses the entire state space. It's two children each encompass a mutually exclusive subset of the state space, and so on. We consider the relationship between the subset of the state space corresponding to a node in the tree and the

vibration space. The vibration space might be outside of the state space, fully contained by the state space, or intersect the boundary of the state space. We posit that if the vibration space intersects the boundary of the node state space, we can use that as a proxy for the policy making a decision based on that vibration. Potentially, the decision it makes not enough to dampen the vibration. In Algorithm 1, we want to find all nodes where the surface of the vibration space intersects with the surface of the node's state subspace. (Note we want to check for intersection of surface of the shapes and not intersection of the volumes of the shape because there will always be at least one node at every depth with contains the vibration space within it's state subspace, even if the vibration space is nowhere near the boundary.

Algorithm 4: Reduce Vibration Method 2

Modify Children of Vibration-Relevant Nodes($\pi^{\dagger}$,$\mathcal{N}_V$,$\mathbf{M}_c$):
for *node* $n_V \in \mathcal{N}_V$ **do**
  if n_V *is a leaf node* **then**
    Let $\mathcal{N}_L$ be a set of nodes containing n_V only;
  else
    Let $\mathcal{N}_L$ be a set of nodes containing all leaf node descendants of n_V;
  end
  for *node* $n_L \in \mathcal{N}_L$ **do**
    n[action] $\leftarrow \mathbf{M}_c[n$[action]];
  end
end
Return the updated $\pi^{\dagger}$

In Algorithm 4, see the procedure for modifying the tree based on the identified nodes. The algorithm makes a modification to the action of all of the leaf nodes descended from the nodes which have boundaries intersecting the vibration space. In general, to reduce vibration, magnitude of velocities is decreased. The specific mapping is provided by $\mathbf{M}_c : T \rightarrow T$, where T is the action type (in our case integer). In our case, we used $\mathbf{M}_c = \{0 \rightarrow 13; 1 \rightarrow 7; 2 \rightarrow 6; 3 \rightarrow 3; 4 \rightarrow 14; 8 \rightarrow 6; 7 \rightarrow 8; 9 \rightarrow 13; 10 \rightarrow 14; 11 \rightarrow 13; 12 \rightarrow 14; 13 \rightarrow 13; 14 \rightarrow 14\}$, with the action space described in Table V.

B. Limitations

MSVIPER requires a discrete action space and a state space in the format of a numerical array. In addition, MSVIPER (like its predecessors) uses hyperparameters that must be tuned for optimal results. These include sample size n_S, number of iterations n, and trajectories generated per iteration m. There may be tasks that a neural net can solve but which requires such a large tree that it is computationally infeasible, such tasks would not be a good fit for MSVIPER or related methods. Finally, since MSVIPER's first step is an imitation process, there is a chance that it needs to deal with the issues of distribution shifting.

In our indoor navigation environment with obstacles, we limited obstacles to be static or dynamic with relatively slow speed, which could be captured by robots' sensors. If the robot cannot sense the obstacle quickly enough, it will not be able to avoid it. One of the benefits of the tree modification is that it allows for incorporating human knowledge. However, it does require a human to encode that knowledge explicitly in a program in order to capture it.

C. MSVIPER Analysis: Full Proofs

In the following sections, we analyze the upper bound on divergence between the tree and the expert policy in the cases of MSVIPER and VIPER, and we compare expectations for sample complexity of VIPER and MSVIPER.

(1) Upper Bound on Divergence Between Expert and Learned Policy

Our work makes the decision tree efficiently and effectively mimic the expert policy. If the loss function between the expert policy and the decision tree is smaller, the decision tree imitates the expert policy better. Therefore if the upper bound of the loss function is smaller, the decision tree would be guaranteed to have a better imitating performance.

Theorem 1 *For the multi-stage MSVIPER, the loss function is :*

$$\tilde{l}_t(s,\pi) = V_t^{(\pi^*)}(s) - \min_{a \in A} \hat{Q}_t^{(\pi^*)}(s,a). \tag{2}$$

where

$$\hat{Q}^{\pi^*}(s,a) = \mathbb{E}_\pi \left[\sum_{e \in E} \left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}^e | S_t = s, A_t = a \right) \cdot w_e \right] \tag{3}$$

Proof According to [4], the loss function is:

$$l(\pi) = T^{-1} \sum_{t=0}^{T-1} \mathbb{E}_{s \sim d_t^{(\pi)}} [l_t(s,\pi)] \tag{4}$$

where T is the time horizon (maximum timesteps) and the loss $l_t(s,\pi)$ is:

$$l_t(s,\pi) = V_t^{\pi^*}(s) - Q_t^{\pi^*}(s,\pi(s)). \tag{5}$$

where $V(s)$ is the value function of the policy and $Q(s,a)$ is the quality function. In order to guarantee a better performance, we need to decrease the upper bound of $\tilde{l}_t(s,\pi)$. After training by training set E we have oracle policy π^*. According to the Q-Dagger algorithm, the hinge loss function $l(\pi)$ is upper bounded by a convex loss $\tilde{l}_t(s,\pi)$:

$$\tilde{l}_t(s,\pi) = V_t^{(\pi^*)}(s) - \min_{a \in A} Q_t^{(\pi^*)}(s,a). \tag{6}$$

When we train the decision tree in multiple environments $e \in E$, considering each environment has a different contribution to policy, we have the weight for each environment e:

$$\omega_e = f(R^e, T^e) \tag{7}$$

Here $\sum_{e \in E} \omega_e = 1$, R_i^e is the reward at timestep i for environment e, and T^e is the total timesteps of trajectories in each scenario e. Then the quality function $\hat{Q}^{\pi^*}(s,a)$ of multiple stages could be:

$$\hat{Q}^{\pi^*}(s,a) = \sum_{e \in E} Q^{\pi^*}(s,a) \cdot \omega_e \tag{8}$$

$$\hat{Q}^{\pi^*}(s,a) = \mathbb{E}_\pi \left[\sum_{e \in E} \left(\sum_{k=0}^{\infty} \gamma^k R^e_{t+k+1} | S_t = s, A_t = a \right) \cdot w_e \right] \tag{9}$$

Replacing $Q^{\pi^*}(s,a)$ with $\hat{Q}^{\pi^*}(s,a)$ we have:

$$\tilde{l}_t(s,\pi) = V_t^{(\pi^*)}(s) - \min_{a \in A} \hat{Q}_t^{(\pi^*)}(s,a). \tag{10}$$

This completes the proof of Theorem 1.

According to [4], while in training, the loss function can be calculated by:

$$Tl(\pi) = J(\pi) - J(\pi^*) \tag{11}$$

where π is any policy trained by the decision tree, and π^* is the policy optimally imitating expert policy. For multiple scenarios, we obtain:

Lemma 1 *Let π^* be held constant, Equation* (12) *holds when using samples from different scenarios.*

$$J(\hat{\pi}) \leq J(\pi^*) + T\epsilon_N + \tilde{O}(1) \tag{12}$$

Proof [4] proposed equation (12) for single stage training. (See [4, 27] for more details on the $\tilde{O}(1)$.) According to [27], for each trajectory in different scenarios $J(\pi) = T\mathbb{E}_{s \sim d_\pi^t}[C_\pi(s)]$ holds. Since sampled trajectories are independent, we can simply add the trajectories of different scenarios into states distribution class D, then the function ϵ_N holds. As a result, the policy $\hat{\pi}$ satisfies equation 12.

As per Lemma 1, the difference between an expert policy and a well-trained policy by MSVIPER is also upper bounded. In the next section we will prove that MSVIPER can decrease the upper bound of a loss function, and thus that MSVIPER has a better performance than VIPER.

(2) Sampling Complexity of MSVIPER

Theorem 2 *$P_M(\epsilon|S_C,l,m,E) \geq P_V(\epsilon|S_C,l,m)$ and $u_M \leq u_V$ (MSVIPER has a sample complexity equivalent or superior to that of VIPER) if $\overline{\mathbb{S}}_k^{e_i,l} < \overline{\mathbb{S}}_k^{e_{n_E},l}$ for any k and $i \neq n_E$.*

We define "sampling complexity" as a measure of how many state-action pair tuples (and thus trajectories generated) are required from the expert policy to learn a decision tree that achieves a given threshold of performance (as measured by average reward achieved compared to the expert policy).

We assert that, given certain assumptions stated in this section, MSVIPER has a sample complexity as good as or better than VIPER. This is shown when the upper bound u for the divergence in reward between the expert policy and the tree policy is, in the MSVIPER case, smaller than or equivalent to the VIPER case.

Sample complexity is improved when the likelihood that high accuracy is achieved on critical states increases.

We look at this upper bound u, described above and in [4] as

$$Q_t^{(\pi^*)}(s,a) - Q_t^{(\pi^*)}(s,\pi^*(s)), \quad \text{for all } a \in A, s \in S, \text{ and } t \in 0,...,T-1. \tag{13}$$

It is noted that u "may be $O(T)$ ...if there are ***critical states*** such that failing to take action $\pi^*(s)$ in state s results in forfeiting all subsequent rewards." This is the case, for example, for the navigation environment, where a collision results in an immediate failure. The states where possible collision is imminent are the "critical states." The same principal can be applied to other domains. In this instance $u = O(T)$. It is further noted that this bound is $O(T)$ "as long as $\hat{\pi}$ achieves high accuracy on critical states." It follows that any improvement to the method that results in increasing the likelihood that high accuracy is achieved on critical states improves the likelihood of the bound being $O(T)$ as a opposed to a higher value.

Let $S_C \in S$ be the set of all critical states. Let $k_i \in S_C$ for $i = 1$ to K be a list of all critical states. We assume that the definition of whether a state is considered a critical state is the same across scenarios. We assume that S_C is a finite set.

Consider the sampling step of our algorithm

$$D_i \leftarrow \{(s_j, \pi^*(s_j)) \sim d^{s_{j-1}}\} \tag{14}$$

Assume that the expert policy π^* chooses correct actions for critical states, and that the decision tree $\hat{\pi}$ could thus only learn to fail on critical states if it fails to sample from critical states during trajectory generation.

Let $\mathbb{S}_k^{e,l}$ be a binary random variable, with the associated probability distribution being the Bernoulli distribution with probability $p = p_k^{e,l}$ (as in, $P(\mathbb{S}_k^{e,l}) = p_k^{e,l}$). (Note $0 \leq p_k^{e,l} \leq 1$). For a given generation of a trajectory of length l drawn from environment e, $(\mathbb{S}_k^{e,l} == 1)$ represents the outcome that the critical state k is sampled as part of the trajectory. $(\mathbb{S}_k^{e,l} == 0)$ represents the outcome that the critical state k is not among the trajectory (although other critical states may be). Let $\overline{\mathbb{S}}_k^{e,l}$ be a similar binary random variable, where an outcome of $(\overline{\mathbb{S}}_k^{e,l} == 1)$ means that state k is not among the states within the drawn trajectory.

Let $P(\epsilon | S_C, l)$ be the probability that the number of critical states present in a drawn trajectory of length l is a fraction of the total number of critical states S_C equal to or greater than ϵ, for $0 < \epsilon < 1$.

The number of critical states drawn from a trajectory of length l from environment e can be represented by the sum $\mathbb{S}_{k_1}^{e,l} + \mathbb{S}_{k_2}^{e,l} + ... + \mathbb{S}_{k_K}^{e,l}$ and the fraction of total states drawn is

$$\frac{\mathbb{S}_{k_1}^{e,l} + \mathbb{S}_{k_2}^{e,l} + ... + \mathbb{S}_{k_K}^{e,l}}{K} \tag{15}$$

For a single environment e,

$$P(\epsilon|S_C,l) = P\left(\frac{\mathbb{S}^{e,l}_{k_1} + \mathbb{S}^{e,l}_{k_2} + ... + \mathbb{S}^{e,l}_{k_K}}{K} \geq \epsilon\right) \tag{16}$$

For single-scenario VIPER using a training that ends up incorporating m total trajectories, the probability of a fraction of the total critical states being $\geq \epsilon$ is

$$\begin{aligned} &P_V(\epsilon|S_C,l,m) = \\ &\quad = P\left(\frac{1}{K}\sum_{k\in S_C}\left(\mathbb{S}^{e,l}_k\right) \geq \epsilon\right) \text{ repeated } m \text{ times} \\ &\quad = P\left(\frac{1}{K}\sum_{k\in S_C}\left(\sum_{j=1}^{m} X_j \sim \mathbf{B}\left(m, p^{e,l}_k\right)\right) \geq \epsilon\right) \end{aligned} \tag{17}$$

where $\mathbf{B}(m,p)$ is the Binomial distribution.

For MSVIPER using a training that ends up incorporating m total trajectories evenly divided across environments $e_1,...e_{n_E} \in E$ (assuming m must be evenly divisible by E), the probability of a fraction of the total critical states being $\geq \epsilon$ is

$$\begin{aligned} &P_M(\epsilon|S_C,l,m,E) = \\ &\quad = P\left(\frac{1}{K}\sum_{k\in S_C}\left(1 - \prod_{i=1}^{n_E}\left(\overline{\mathbb{S}}^{e_i,l}_k\right)\right) \geq \epsilon\right) \text{ repeated } m \text{ times} \\ &\quad = P\left(\frac{1}{K}\sum_{k\in S_C}\left(\sum_{j=1}^{m} X_j \sim \mathbb{B}\right) \geq \epsilon\right), \\ &\qquad \mathbb{B} = \mathbf{B}\left(\frac{m}{n_E}, 1 - \left(p^{e_1,l}_k \cdot p^{e_2,l}_k \cdot .... \cdot p^{e_{n_E},l}_k\right)\right) \end{aligned} \tag{18}$$

Note that $P_V(\epsilon|S_C,l,m) = P_M(\epsilon|S_C,l,m,E)$ if E consists only of e_{n_E}.

If any $\overline{\mathbb{S}}_k^{e_i,l} < \overline{\mathbb{S}}_k^{e_{n_E},l}$ for any k, for any $i \neq n_E$ (where e_{n_E} indicates the final stage) then we can say that

$$P_M(\epsilon|S_C, l, m, E) \geq P_V(\epsilon|S_C, l, m) \tag{19}$$

Given this relationship between the probabilities, we can say that for any ϵ where the assumptions noted above hold, the probability that MSVIPER utilizes this ϵ-fraction of critical states in its training of the tree is greater than or equal to the probability that VIPER does so. Thus, under these conditions the upper bounds u for the divergence between expert and tree are equivalent between VIPER and MSVIPER, or MSVIPER has a smaller bound. $(u_M \leq u_V)$ Thus, MSVIPER has equivalent or superior sample complexity, provided that any $\overline{\mathbb{S}}_k^{e_i,l} < \overline{\mathbb{S}}_k^{e_{n_E},l}$ for any k and $i \neq n_E$. This completes the proof of Theorem 2.

AARON M. ROTH is the Head of Autonomy Technology at Black Sea Technologies. He has conducted research in robotics and AI at the University of Maryland, Carnegie Mellon University, and the Naval Research laboratory. Aaron Roth's website is www.aaronmroth.com.

JING LIANG received the M.S. degree in robotics from the University of Maryland, College Park, MD, USA, in 2020, where he is currently pursuing a Ph.D. degree in computer science. he worked at Clever Sys Inc., WaterMirror Technology, and Amazon Lab126. His research interest is Robotic Navigation.

RAM D. SRIRAM is the president-elect of the Washington Academy of Sciences and is currently the chief of the Software and Systems Division, Information Technology Laboratory, at the National Institute of Standards and Technology. Sriram has received numerous awards and is a Fellow of AAIA, AIBME, ASME, AAAS, IEEE, IET, INCOSE, SMA, and the Washington Academy of Sciences.

ELHAM TABASSI is a Senior Research Scientist at the National Institute of Standards and Technology (NIST) and the Associate Director for Emerging Technologies in the Information Technology Laboratory (ITL). She is a member of the National AI Resource Research Task Force, vice-chair of OECD working party on AI Governance, Associate Editor of IEEE Transaction on Information Forensics and Security, and a fellow of Washington Academy of Sciences. In 2023, TIME magazine has named Elham Tabassi in its list of the 100 most influential people in AI.

DINESH MANOCHA is Paul Chrisman-Iribe Chair in Computer Science & ECE and a Distinguished University Professor at the University of Maryland College Park. His research interests include virtual environments, physically-based modeling, and robotics. He is a Fellow of AAAI, AAAS, ACM, IEEE, and NAIm member of ACM SIGGRAPH Academy, and Bézier Award recipient from Solid Modeling Association. He received the Distinguished Alumni Award from IIT Delhi and Distinguished Career Award from Washington Academy of Sciences

Albert H. Teich

Albert H. Teich died April 26, 2022, after a fall and ten days in intensive care in N. Bethesda, MD. He received his B.S. in Physics (1964) and Ph.D. in Political Science (1969) from MIT.

Professionally, Al was involved with science policy issues such as visa and immigration policy, research and development in the federal budget, climate change, and human rights. He was most currently a Research Professor of Science, Technology & International Affairs at the Institute for International Science and Technology Policy in the Elliott School of George Washington University in Washington, DC. Prior to joining the GW faculty, he was director of Science & Policy Programs for the American Association for the Advancement of Science (AAAS), where he was responsible for the Association's activities in science and technology policy for 32 years, managed a staff of about 40 people, and served as a key spokesman on science policy issues. He published Technology and the Future (originally Technology and Man's Future) in 1971, revising the compilation of essays through 12 editions, ending in 2012, and it remains in use as a coursebook of readings in science and technology issues.

While at AAAS he contributed to numerous articles and books, made innumerable presentations, testified multiple times before Congressional committees, and participated on advisory committees in the U.S. and around the world. He became a Fellow of AAAS in 1986, received an Award for Scientific Achievement in Science Policy from the Washington Academy of Sciences in 2004, and became an honorary member of the Washington Science Diplomats Club the same year. He was a constant presence in Washington science policy circles, and he was frequently sought after as a commentator and advisor on issues related to "science and technology policy" and "science in society."

This is an excerpt from the full obituary at https://bit.ly/3uRWNup.

Delegates to the Washington Academy of Sciences Representing Affiliated Scientific Societies

Acoustical Society of America	Paul Arveson
American/International Association of Dental Research	J. Terrell Hoffeld
American Association of Physics Teachers, Chesapeake Section	Frank R. Haig, S. J.
American Astronomical Society	Sethanne Howard
American Fisheries Society	Lee Benaka
American Institute of Aeronautics and Astronautics	David W. Brandt
American Institute of Mining, Metallurgy & Exploration	E. Lee Bray
American Meteorological Society	Vacant
American Nuclear Society	Charles Martin
American Phytopathological Society	Vacant
American Society for Cybernetics	Stuart Umpleby
American Society for Microbiology	Vacant
American Society of Civil Engineers	Vacant
American Society of Mechanical Engineers	Daniel J. Vavrick
American Society of Plant Physiology	Mark Holland
Anthropological Society of Washington	Vacant
ASM International	Toni Marechaux
Association for Women in Science	Jodi Wesemann
Association for Computing Machinery	Vacant
Association for Science, Technology, and Innovation	F. Douglas Witherspoon
Association of Information Technology Professionals	Vacant
Biological Society of Washington	Vacant
Botanical Society of Washington	Chris Puttock
Capital Area Food Protection Association	Keith Lempel
Chemical Society of Washington	Vacant
District of Columbia Institute of Chemists	Vacant
Eastern Sociological Society	Ronald W. Mandersheid
Electrochemical Society	Vacant
Entomological Society of Washington	Vacant
Geological Society of Washington	Jeff Plescia Jurate Landwehr
Historical Society of Washington DC	Vacant
Human Factors and Ergonomics Society	Gerald Krueger

(continued on next page)

Delegates to the Washington Academy of Sciences Representing Affiliated Scientific Societies

(continued from previous page)

Institute of Electrical and Electronics Engineers, Washington Section	Richard Hill
Institute of Food Technologies, Washington DC Section	Taylor Wallace
Institute of Industrial Engineers, National Capital Chapter	Neal F. Schmeidler
International Association for Dental Research, American Section	Christopher Fox
International Society for the Systems Sciences	Vacant
International Society of Automation, Baltimore Washington Section	Richard Sommerfield
Instrument Society of America	Hank Hegner
Marine Technology Society	Jake Sobin
Maryland Native Plant Society	Vacant
Mathematical Association of America, Maryland-District of Columbia-Virginia Section	John Hamman
Medical Society of the District of Columbia	Julian Craig
National Capital Area Skeptics	Vacant
National Capital Astronomers	Jay H. Miller
National Geographic Society	Vacant
Optical Society of America, National Capital Section	Jim Heaney
Pest Science Society of America	Vacant
Philosophical Society of Washington	Michael P. Cohen
Society for Experimental Biology and Medicine	Vacant
Society of American Foresters, National Capital Society	Marilyn Buford
Society of American Military Engineers, Washington DC Post	Vacant
Society of Manufacturing Engineers, Washington DC Chapter	Vacant
Society of Mining, Metallurgy, and Exploration, Inc., Washington DC Section	E. Lee Bray
Soil and Water Conservation Society, National Capital Chapter	Erika Larsen
Technology Transfer Society, Washington Area Chapter	Richard Leshuk
Virginia Native Plant Society, Potomac Chapter	Alan Ford
Washington DC Chapter of the Institute for Operations Research and the Management Sciences (WINFORMS)	Meagan Pitluck-Schmitt
Washington Evolutionary Systems Society	Vacant
Washington History of Science Club	Albert G. Gluckman
Washington Paint Technology Group	Vacant
Washington Society of Engineers	Alvin Reiner
Washington Society for the History of Medicine	Alain Touwaide
Washington Statistical Society	Michael P. Cohen
World Future Society, National Capital Region Chapter	Jim Honig

Washington Academy of Sciences
Room 455
1200 New York Ave. NW
Washington, DC 20005
Return Postage Guaranteed

NONPROFIT ORG
US POSTAGE PAID
MERRIFIELD VA 22081
PERMIT# 888

4*7*********109**********AUTO**MIXED ADC 220
HARVARD LAW S LIB ERSMCZ
LANGDELL HALL 152
1545 MASSACHUSETTS AVE
CAMBRIDGE, MA 02138-2903